Semantic Web Technologies for Method Management

Model, Methodology and Enterprise Architecture Integration

Peter Christian Rosina

Dissertation

for the degree of

Doctor of Natural Sciences (Dr. rer. nat.)

University of Augsburg

Institute of Computer Science

Software Methodologies
for Distributed Systems

December 2015

Semantic Web Technologies for Method Management
Model, Methodology and Enterprise Architecture Integration

Supervisor:	**Prof. Dr. Bernhard Bauer**, Institute of Computer Science, University of Augsburg, Germany
Advisor:	**Prof. Dr. Elisabeth André**, Institute of Computer Science, University of Augsburg, Germany
Thesis Defense:	February 15th, 2016

Abstract

In order to face today's challenges in product development, like product complexity, variability and a shortening of development cycles, for instance, in the automotive domain, the Product Development Process (PDP) tends to entail more and more virtual tasks instead of physical, conventional ones, for example, implemented by Computer Aided *x* (CAx) technologies. The here introduced approach promotes this transformation by supporting the appropriate stakeholders in acquiring, formalizing, analyzing, assessing, comparing and hence selecting the most suitable methods, like physical and virtual design methods or CAx methods, that are utilized to execute these tasks.

The introduced methodology demonstrates the acquisition and integration of relevant domain and business knowledge from diverse enterprise knowledge sources using Semantic Web Technologies (SWTs) which facilitates the management and execution of this knowledge, while considering customized individual views and vocabularies, and, as a result, enables a higher flexibility regarding and faster reaction to the ever-changing PDP by supporting stakeholders in their strategical and operational decisions. The separation of knowledge types by applying SWTs, like ontologies, rules and queries, enables the appropriate roles to manage this knowledge independently and in their own way, because (conceptual) domain knowledge, (operational) business rules and an implemented data model have different lifecycles, scopes and owners. Furthermore, this particular domain and business knowledge features interdependencies with remaining enterprise knowledge, including business processes, organizational aspects and the IT architecture, which is usually modeled in an Enterprise Architecture (EA).

Therefore, we showcase the implementation and integration of a method meta model, method contexts, like metrics and product knowledge, and an EA using SWTs to promote the transparency, interchange, interconnection, synchronization, sharing, reusability, re-deployment, up-to-dateness and maintenance of this particular knowledge and consequently analyses and assessments based on it. Thus, the use of SWTs increase the flexibility, quality and efficiency of the development process, allowing enterprises to meet the increasing market demand of product diversification, increasing functional complexity and regulatory requirements, for instance, due to an improved integration of virtual development.

Acknowledgements

First and foremost, I am very grateful for the dedicated and continuous support, encouragement, feedback and guidance by my supervisor Prof. Dr. Bernhard Bauer during the course of this thesis. He helped me to structure my work, to focus on the important parts, urged me to publish my results while granting me great liberties during all phases of my doctoral studies. Besides, he responds to emails in a jiffy and always takes his time for discussions.
Furthermore, I want to thank my reviewer Prof. Dr. Elisabeth André for her supervision, effort and opinion.
Next to my professors at the University of Augsburg, Audi AG has been an important sponsor and initiator of my dissertation. Therefore, I am grateful to the management for giving me the chance to participate in their doctoral candidate program and my colleagues there for fruitful discussions, feedback and collaboration, predominately while developing and evaluating the case studies. Most important, Thomas Syldatke, my mentor at Audi during my time there, has always been supportive, committed and promoted my ideas, for example, when defining the use cases, conducting the evaluation or explaining topics concerning the enterprise, method or product development. Through him, I could participate in the ONTORULE project[1], where I had the opportunity to meet, cooperate with and learn from experts in the numerous applied disciplines in this thesis, for instance, ontologies, rules and knowledge acquisition.
A big thank you to my colleagues at the SMDS chair for their professional feedback, companionship and lots of humor.
Last but not least, I am very grateful to my friends and family for their support and motivation.

[1]The ONTORULE project was an EU FP7 project, partially funded by the European Commission under Grant Agreement n° 231875. http://ontorule-project.eu/; last accessed 11/25/2015.

Contents

I. Introduction, Basics and Related Work **1**

1. Introduction **3**
- 1.1. Introduction and Motivation . . . 3
- 1.2. Challenges, Objectives and Approaches . . . 10
  - 1.2.1. Challenges . . . 10
  - 1.2.2. Objectives . . . 15
  - 1.2.3. Approaches . . . 18
- 1.3. Publications . . . 24
- 1.4. Outline . . . 27

2. Basics **31**
- 2.1. Knowledge Management . . . 31
- 2.2. Semantic Web . . . 34
- 2.3. Ontologies . . . 36
  - 2.3.1. Foundations . . . 37
  - 2.3.2. Description Logics . . . 40
  - 2.3.3. Ontology Languages and Standards . . . 43
- 2.4. Rules . . . 49
  - 2.4.1. Rule Languages and Standards . . . 52
  - 2.4.2. Rule Engines and Management Software . . . 53
- 2.5. ONTORULE Methodology . . . 54
- 2.6. Virtual Product Development . . . 55
- 2.7. Enterprise Architecture Management . . . 59
  - 2.7.1. Enterprise Architecture Frameworks . . . 62
  - 2.7.2. The Open Group Architecture Framework . . . 63
  - 2.7.3. ArchiMate . . . 65

3. Related Work **69**
- 3.1. Synopsis . . . 69
- 3.2. Method Frameworks and Meta Models . . . 70
- 3.3. Method Metrics . . . 77

3.4. Modeling Standards 78
3.5. Integrating method knowledge in an Enterprise Architecture 80
3.6. Views 84
3.7. Method Selection, Analysis and Application 84
3.8. Comparative Assessment 89

II. METHOD META MODEL, METHOD ONTOLOGY, METHODOLOGY AND ENTERPRISE ARCHITECTURE INTEGRATION 95

4. Method Meta Model and Ontology 97
4.1. Introduction 97
4.2. Method Definition 101
4.2.1. Definition 103
4.2.2. Methods vs. Processes 109
4.3. The Core Method Ontology 113
4.3.1. The Ontology Concepts in Detail 114
4.3.2. Representation Models of Methods 116
4.4. Method Metrics 119
4.4.1. Method Maturity 119
4.4.2. Concrete vs. Abstract Methods 122
4.4.3. Metrics Ontology 124
4.5. Linking Enterprise Data 126
4.5.1. Ontology Matching Strategies 127
4.5.2. Using Rules for Ontology Alignment 132
4.5.3. Integrating Databases 134
4.6. Integrating the Method Ontologies 137
4.6.1. Resources 139
4.6.2. Method Description 143
4.6.3. Using the Method Ontology as a Controlled Vocabulary . . . 145
4.7. Conclusion 149

5. Methodology 153
5.1. Introduction 153
5.2. Approach 154
5.3. Conclusion 159

6. Enterprise Architecture Integration **161**
6.1. Introduction . . . 161
6.2. Integration of the Method Model . . . 162
6.2.1. EA Ontology . . . 162
6.2.2. Mapping Techniques for the Integration of the EA and Method Ontologies . . . 164
6.2.3. Mapping the EA and Core Method Ontologies . . . 166
6.2.4. Further Mappings . . . 168
6.3. Views and Roles . . . 170
6.3.1. Introduction . . . 170
6.3.2. Stakeholders . . . 172
6.3.3. Creating Views . . . 176
6.4. Method Analysis and Selection . . . 178
6.4.1. Method Classification . . . 179
6.4.2. Method Selection . . . 180
6.4.3. Method Architecture Artifacts . . . 183
6.4.4. Supported Business Goals . . . 192
6.5. Conclusion . . . 194

III. EVALUATION AND CONCLUSIONS **197**

7. Case Studies **199**
7.1. Synopsis . . . 199
7.2. Knowledge Management with a lexicalized ontology and rules . . . 201
7.2.1. Introduction . . . 201
7.2.2. Use case background . . . 203
7.2.3. Expressing and linking the vocabulary . . . 204
7.2.4. Conclusion . . . 210
7.3. Semantic Integration of Bill of Materials . . . 212
7.3.1. Introduction . . . 212
7.3.2. Bill of Materials . . . 215
7.3.3. Approach . . . 218
7.3.4. Implementation . . . 224
7.3.5. Evaluation . . . 229
7.3.6. Conclusion . . . 231
7.4. Aligning the Method Model with Industrial Standards . . . 234

7.5. Comparing Methods in Property-Driven Development 238
7.5.1. Introduction 238
7.5.2. Property-Driven Development 240
7.5.3. Approach 246
7.5.4. Implementation 251
7.5.5. Evaluation 259
7.5.6. Conclusion 265

8. Conclusion **269**
8.1. Contributions 269
8.2. Outlook 273

IV. ANNEX **277**

Printed Sources **279**

Online Sources **297**

Glossary **303**

List of Figures **309**

List of Tables **313**

Listings **315**

A. BOM Mapping Case Study **317**
A.1. Ontologies 317
A.2. Rules and Queries 319

B. PDD Case Study **321**
B.1. User Interface 321
B.2. Rules and Queries 325
B.3. Ontology Schemata 327

Part I.

Introduction, Basics and Related Work

"Begin at the beginning," the King said, very gravely, *"and go on till you come to the end: then stop."*

Lewis Carroll, *Alice in Wonderland*

1 Introduction

1.1. Introduction and Motivation

During the development of a new industrial product, for instance, a car, it is undergoing a multitude of different processes and process steps, e.g., product planning and design in different stages, before a manufacturer can finally start its production in series. Naturally, a multitude of different departments and disciplines, for instance, physical tests and virtual simulations, have to collaborate, cooperate and interact during this Product Development Process (PDP) which entails the application of numerous technologies, tools and methods, for instance, Computer Aided x (CAx) methods. A wide range of CAx methods, such as verification methods, plausibility checks or Finite Element Methods (FEMs), are applied or developed throughout the entire PDP. In many other disciplines, to prevent the loss of track about the already acquired knowledge and insights, the current status and the planned procedures, a plethora of monitoring, planning, information, knowledge and Product Data Management (PDM) systems are in operation. Among others, they can keep track of processes, finances, Information Technology (IT) or resources, like drawings or parts, because these domains are well understood and in many cases standardized.

However, an eagerly considered area in the Knowledge Management (KM) of car manufacturers has been the difficult exchange, analysis and sharing of CAx methods and their contexts within and between the various Computer Aided (CA) disciplines. For example, CAx comprises the Computer Aided Design (CAD) that provides approaches for the virtual design and verification of products and their geometry, e.g., Digital MockUp (DMU) methods or parametric design methods. Further involved disciplines are the Computer Aided Engineering (CAE) and Computer Aided Testing (CAT); the former provides methodologies for simulating the behavior of a product and its functions, e.g., FEM for crash simulation, Computational Fluid Dynamics (CFD) for thermal management, and Multi Body Simulation (MBS) for driving dynamics; the latter for performing physical tests, e.g., vehicle management, job, testing control and test result analysis. The involved tools and methods range from mechanical test beds, through Hardware in the Loop (HiL) to DMU and other pure software applications.
Unfortunately, it is not possible to compare, interchange, share or consolidate these different methods innately, because they lack a consistent description, terminology and harmonized semantics. Even inside their respective CAx domain, their structure or vocabulary differs from each other dependent on the involved stakeholders, roles, departments, processes, IT applications or the data and resources they require or produce.
Besides, this challenge does not only affect CAx methods in the automotive development, but all the applied design methods during a PDP for complex and sophisticated products in various markets, such as aeronautical, mechanical and civil engineering or "traffic, energy, medicine, and the environment" (Meerkamm 2011). "The development of these products needs a well-balanced and integrated use of design methods and computer-supported [(CA)] tools. Therefore, bringing together both fields is a very important task, needing integration on a methodological basis of methods with tools and tools with the process" (ibid.).
This challenge is aggravated by the fact that "globalization has increased competitive pressure on all companies, and time to market has become an essential success factor" (Binz et al. 2011). Furthermore, the entire development and its contexts become more complex, e.g., owing to internal factors like products, processes and organization or external ones like markets, norms and regulations (Ehrlenspiel and Meerkamm 2013), which is another challenge to master for today's industries. One reason for this complexity is the increased diversity of variants which is partially justified due to more individualized products and systems.

For example, the amount of different vehicle models regarding the top five brands in Germany has more than quadrupled (from 101 to 453) between 1990 and 2014 (Progrenium 2015). In addition, the masses of possible features and feature combinations a customer can choose from today has amplified manifoldly.

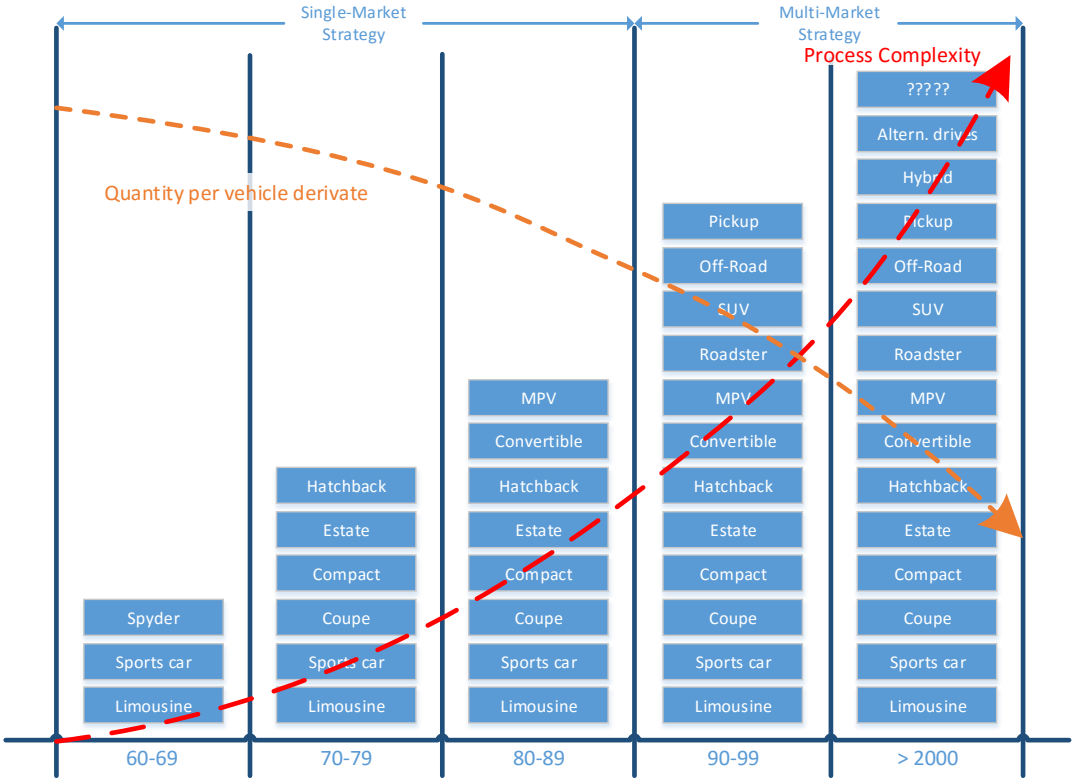


Figure 1.1.: Development of the product strategy in the automotive industry. Based on Pätzold, ProSTEP AG, 2002 (obtained from Eigner and Stelzer (2009a)).

The trend for a broader product range started several decades ago and is still increasing as depicted in Figure 1.1, especially when considerung multi-market demands, strategies and regulations. However, when looking at the German market and the top five brands again, the number of new registrations stagnates at about 3 million per year (ibid.). That implies, that on the one hand, the quantity per derivative is decreasing but on the other hand, the process complexity is rising. Moreover, car manufacturers expect the demand for passenger cars to rise worldwide by almost 60% over the next 15 years due to a global growth in prosperity (Stadler 2011). "And the premium segment will even grow faster than

the market as a whole. [...] Admittedly these figures represent an enormous increase in complexity. And a major challenge for the entire organization" (Stadler 2011). This complexity of a car model is exemplified in Figure 1.2, representing the coherences between loading cases, for instance, crash, vibration comfort or pedestrian protection, and assemblies and components of the "Body in White" (Ehrlenspiel and Meerkamm 2013) in the product development at the car manufacturer Audi. The product development is a phase in the PDP which is again a part of the whole product lifecycle (Eigner and Stelzer 2009c).

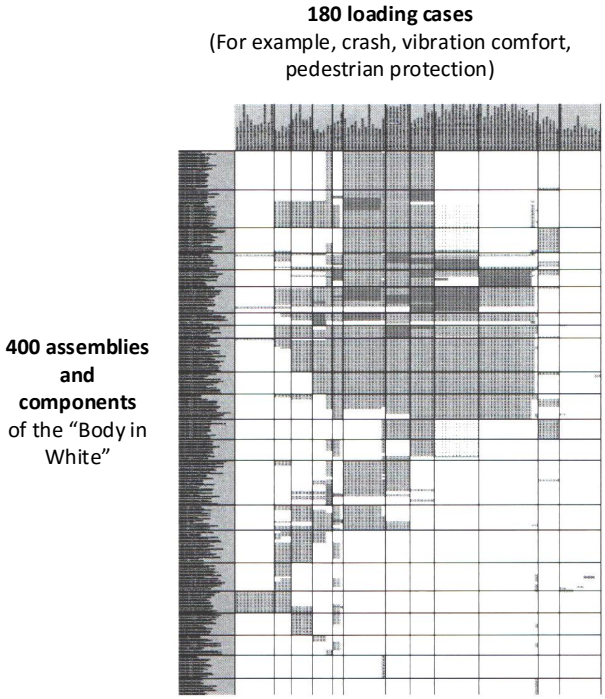


Figure 1.2.: Numerous dependencies between body components and loading cases illustrating the complexity in product development. Based on Ehrlenspiel and Meerkamm (2013).

As illustrated, these complex coherences are hard to handle, visualize and hence, it is no simple task to analyze their interdependencies, especially when trying to imagine the relationships between further linked concepts, like methods that apply these loading cases using components or assemblies as input and producing output resources, either virtually or physically, and, for instance, related process actions, involved user roles and their influence on organizational aspects, like the enterprise' strategical orientation and requirements concerning the supporting IT.

Summarized, enterprises, especially in Research & Development (R&D), have to deal with growing product diversification, increasing functional complexity, changing market demands and tightening regulatory requirements, that have to be taken into account while maintaining a high quality and the costs of development and product as low as possible.
In order to cope with these challenges, the current competitive situation is characterized by shortening development and innovation cycles in addition to a higher innovation rate (Weigt 2008, p. 29). As a consequence, development processes have to be streamlined, loops avoided and results, insights, methods and parts reused in other projects, i.e., "it is necessary to focus on the most promising projects and to take the 'right' decisions" (Binz et al. 2011). However, the PDPs have "to become 'better' in terms of quality and reliability of the development results [because shortening] development times holds the risk of increasing failures, as, for example, in the automotive industry in the past." (ibid.). Besides, today's customers demand higher quality products, e.g., they have to be ecofriendly, safe and secure at low cost, and thus, "it is not only necessary to do 'the right things', but also to do 'things right'" (ibid.). Thereby, customer demands and innovation compulsion cause conflicting goals between *lower costs*, *shortened time consumption* and a *higher product quality* (Weigt 2008, p. 29) which are the main success factors for products (Binz et al. 2011). "Besides the quality of the [PDP], the performance of the applied design methods and tools is decisive in achieving good results" (ibid.).
However, the problem is not a lack of available methods. Instead, it is rather the selection of the most suitable method from a plethora of methods available to the designer (Ernzer and Birkhofer 2002). For example, Domain Experts (DEs) often select popular methods instead of analyzing the real needs of the business (ibid.). Furthermore, methods are not well accepted or regularly used in practice, because the know-how, on how to integrate method knowledge into the PDP and the daily development practice is often not available (Albers, Reiß, Bursac, Urbanec, et al. 2014). Further hindrances are the unevenly distributed knowledge concerning methods, the unawareness of their availability and that only a portion of methods are widely and systematically used. On top of that, applied methods are often used with ad-hoc modifications, many DEs believe them only useful when there is an abundance of time available and they are abandoned after a while (López-Mesa 2003). Nevertheless, if applied correctly, their utility has been demonstrated (Albers, Reiß, Bursac, Urbanec, et al. 2014).

Nowadays, a method selection for a defined purpose, an analysis of the entirety of methods in a company and the method monitoring are mainly manual tasks, ever and anon assisted by basic IT documents, like spreadsheets. Method knowledge is seldom formalized which impedes an IT-supported analysis and thus a profound decision for the application or development of particular methods. Therefore, finding and selecting suitable methods is extremely complex and difficult for designers and other DEs in practice (Badke-Schaub, Daalhuizen, and Roozenburg 2011).
Furthermore, a targeted development, refactoring or shutdown of methods based on business information, like strategical goals or reuse in other processes, is complicated. Formalizing this knowledge would allow the involved roles to analyze methods along the PDP, e.g., for finding, designing and boosting the application of virtual replacements for physical tests. Virtual methods are usually less time-consuming and costly. Hence, they can be repeated many times in controlled environments, i.e., they promote the integration and application of virtual engineering at a maintainable or even superior quality, for example, in order to increase early product maturity.
Other roles, like enterprise architects, would benefit from such a formalization because it allows analyzing the relationships between methods, tools and the underlying IT infrastructure which is usually modeled in an Enterprise Architecture (EA) inside large-scale enterprises. The methodology of introducing, modeling and analyzing (meta) models describing the company's business and IT is conducted in the discipline known as Enterprise Architecture Management (EAM). EA experts and DEs could benefit mutually by combining existing information with method knowledge, because new opportunities for the product, business and IT analysis would emerge. Managers in the enterprise can also benefit from modeled methods and analyses based on these models, because they can help to assess qualities, maturities and further Key Performance Indicators (KPIs) that are influenced by methods which in turn enables an instrument that supports governing and facilitates new strategic decision-making processes. Moreover, stakeholders, like business rules or legal experts, that own knowledge about mandatory regulations that have to be regarded in different markets concerning product tests and design methods could formalize their constraints which in turn influences the selection made by the product developing DEs that are a determining factor in a manufacturing enterprise. "Above all, their knowledge, expertise, competencies and skills are decisive in creating excellent products. For

this reason, it is extremely important to have the 'right' knowledge in a company, to expand, preserve and transfer it to future generations and to make it easily accessible to everybody who needs it. Considering the ongoing explosion in knowledge, the significance of [KM] will increase" (Binz et al. 2011).
However, developing an IT application that is based on, provides and allows interacting with formalized knowledge is no simple task. First of all, the Knowledge Acquisition (KA) and KM in general have to be integrated into a methodology that allows the right roles to interact with and maintain their knowledge independently. Business rules, conceptual domain knowledge, process knowledge and of course the application logic have different areas of competence, owners and lifecycles as exemplified in Figure 1.3. "Domain knowledge is typically long-lived and characterized by a slow evolution. It can be shared among organizations, because it is not organization- or application-specific, but domain-specific. On the other side, operational rules are short-lived and evolve fast, and may be business-specific (e.g., regulations) or particular to an organization (e.g., price discount rules)" (Berrueta et al. 2011).

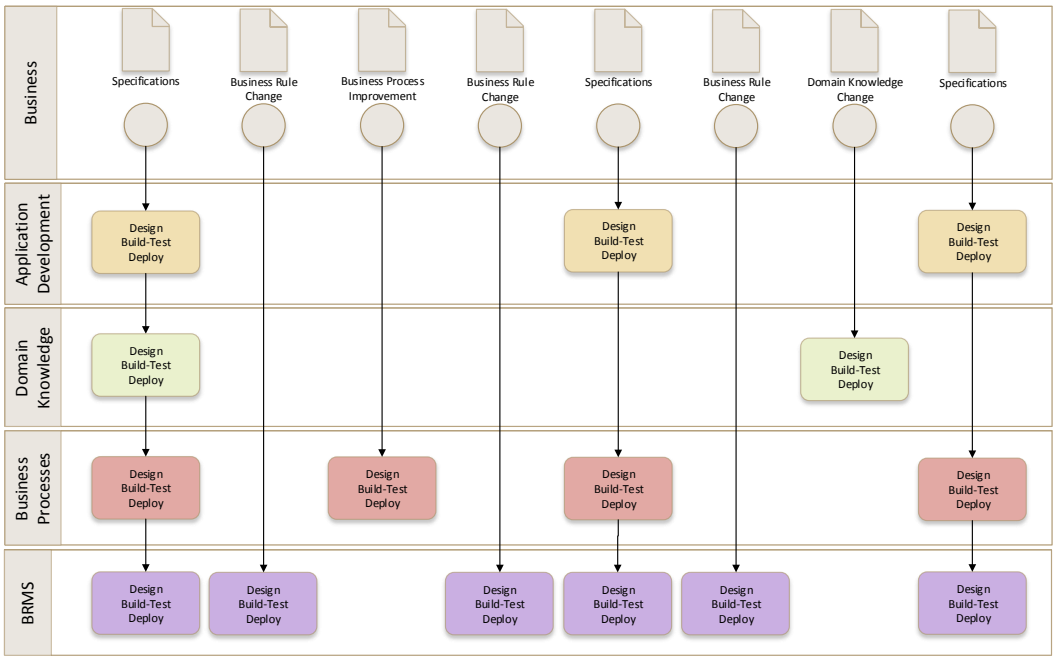


Figure 1.3.: Lifecycle juxtaposition of business, domain knowledge, processes, rules and applications. Based on Businessrules.ch (2008).

Furthermore, applications need to be developed, maintained, redesigned and deployed in separate lifecycles. This way, the particular roles can react faster to

occurring changes, for example, when maintaining their business rules in a Business Rule Management System (BRMS).
Additionally, the different roles work most efficiently when presented with a customized view on their domain and area of competence, i.e., "in order to really cope with the current and future challenges, a conceptual model of the enterprise is needed that is coherent, comprehensive, consistent, and concise, and that only shows the essence of the operation of an enterprise model" (Dietz 2006). Next to explicating unstructured or implicit knowledge, such a KM methodology also needs to incorporate already available (semi-)formalized sources, e.g., databases, in order to reuse as much information as possible. Therefore, the procedure of mapping other Knowledge Bases (KBs) needs to be taken into account, as well.

1.2. Challenges, Objectives and Approaches

1.2.1. Challenges

Challenge 1: Method Integration

Because the method environment is very heterogeneous, for example, the different CAx disciplines, and numerous stakeholders are involved, it is a challenge to express the particular knowledge, structure, contexts and varying definitions in a harmonized way. Furthermore, the involved business units and stakeholders often use their own differing vocabulary for the description of the same entities. Therefore, it is difficult to share, interchange, consolidate, analyze or compare the various methods and their contexts. However, due to factors like a reduced time to market, globalization and an increased product variability these aspects become necessary. Especially the integration of methods and tools and the integration of tools and processes on a methodological basis is a requirement for the successful partnership of Design Methodology and CAx tools (Meerkamm 2011). That means, these concepts need to be clearly defined, because each involves input and output resources and their connected entities model a control flow graph as in processes, tool or method chains.
Another challenge is the integration of various rules, for instance, business rules, into our model. Specific methods are executed to comply with prescribed scenar-

ios, including group, brand, national and international regulations. The knowledge about their mandatory and optional applicability has to be formalized and be compatible with the method meta model. Other rules express how various KPI can be inferred.
Additionally, the system and method information are very dynamic, because new methods are explored and developed all the time. The DEs must be able to enter, remove or modify this information themselves, without the need to bother an IT expert that would have to change application code, because releasing a new software version in a big company is usually a time-consuming process. Besides, the elaborated business knowledge and insights have to be preserved in a standardized format to foster their sustainability, so it can be reused in other projects.
For that reason, an additional requirement for the applied modeling technique is the support for the separation of source code, domain and business knowledge. They have different life cycles and responsibilities, but are strongly interconnected. However, all three aspects ought to be synchronized and act in concert efficiently.

Challenge 2: **Methodology**

Integrating the methods and their environment requires a methodology that supports the KA, formalization, modeling and execution of the domain knowledge, extended with further KBs, business knowledge and logic. The resulting integrated model needs to depict this domain of methods, technologies, tools and processes in order to support stakeholders in their daily work.
A considerable amount of data is spread throughout the company, but it needs to be quickly accessible by the right people in a standardized way in order to be informative. Sometimes, knowledge about whole domains, regulations, methods or IT systems is written down in unstructured or semi-structured textual documents in arbitrary locations. The KA, where present knowledge is formalized and captured from sundry unstructured sources, like documents, spreadsheets or presentations, is a first step into developing such a model. Yet even more precise and extensive knowledge about a domain is only known to a handful of DEs, who are often occupied answering and discussing banalities (from their point of view) instead of doing the important research and development that pushes themselves and the company forward. Therefore, next to transforming knowledge from document-centric sources to model-centric formats, domain models also need to be created by interviewing and analyzing the DEs themselves, while

being as non-intrusive as possible. Finally, knowledge can also be acquired by reusing existing structured data, like databases or available ontologies and business rules (BRs). Besides capturing domain and world knowledge and mapping it into suitable structures, these structures have to be designed correctly and efficiently to guarantee its quality, reusability and performance.
As mentioned in the first challenge, an important obstacle in the collaboration is the use of role-specific terms for the concepts. This leads to difficulties and misunderstandings when two worlds try to discuss and exchange their knowledge. This is not only a problem when talking to each other, since the already formalized business knowledge that exists in databases or other arbitrary places also differs in its representation and scope. Next to the KA, the methodology needs to support knowledge authoring, i.e., the modeling itself, followed by the execution of knowledge.
In classical software engineering, for example, when using the waterfall model, specifying the application's data model after the initial requirements analysis is a widely used procedure when developing a new software. This procedure, however, can lead to negative consequences. On the one hand, the domain and semantic knowledge is part of the conceptual data model and parts cannot be transferred to the logical data model which depends on the implementation (Polikoff 2011). On the other hand, the domain knowledge, that does not fit into the data model, is mixed up with the software's operational rules. This impedes its reuse, sharing, maintenance and evolution. Among other things, because of different lifecycles and responsibilities. This way, an incorrect role, for instance, an IT specialist, would be implementing and maintaining a DE's knowledge.
Besides, different business roles, e.g., IT experts, DEs or managers, work most efficiently when presented with a customized view on the data. For example, the management's view on the KB is a high level view with aggregated data and preferably an excerpt that displays only the important and urgent information. Therefore, the methodology needs to support presenting the logic, domain and business knowledge to the correct roles in their preferred way.

Challenge 3: **Enterprise Architecture and Method Meta Model Integration**

CAx and other design methods feature many dependencies toward existing business and IT information. On the one hand, they almost always require resources, like results from former analyses, physical parts, CAD drawings or assessments

by DEs. For example, the date of their availability is often noted in the company's business processes. On the other hand, successfully performed methods also produce resources like the previously mentioned ones and DEs know how long it takes to perform such a method, including its set-up time, the model creation, the analysis and the following assessment. These indicators allow estimating the time span when a specific method can and has to be performed, including its preceding or succeeding buffers. Furthermore, CA methods, especially virtual methods, heavily depend on software tools and the underlying IT infrastructure. From a business management perspective, the enterprise's general method knowledge, the methods' qualities, costs, time consumptions and maturity are influential factors for the enterprises overall capability, maturity and further KPIs.
Therefore, integrating the method meta model into an Enterprise Architecture Framework (EAF)'s meta model bears many advantages. First of all, the combined meta models allow stakeholders to perform novel kinds of analyses, like impact analyses or discovering business, method and IT relations. For example, the concern "Which system/application supports which methods?" or the responsibility of the modeled actors and roles could be identified. Furthermore, a company benefits from such a mapping approach through a concerted and defined meaning of the modeled concepts and vocabulary.
Additionally, embedding the method meta model into an EA meta model, and therefore making use of existing EAM processes ensures their up-to-dateness and allows the monitoring and analysis in combination with the afore-mentioned business information. Another key aspect of EAM is the planning and migration from a baseline to a target EA which could also be a powerful device, when fostering a strategic method development or shutdown, related to the enterprise's strategic goals. Besides, information about IT applications, processes, roles and organizational aspects can be combined with method knowledge which enables an even more powerful analysis.

Challenge 4: **Method Analysis and Selection**

Virtual methods are constantly evolving and getting more complex. Therefore, it is necessary to have a system that helps managers and DEs to assess the virtual methods concerning their maturity and purpose to help them decide for and select a suitable method or method combination. This approach has to be flexible enough to include information about nearly all kind of development

methods from the involved domains, such as automotive dynamics, acoustics, safety&security, energy management, aerodynamics, 'Noise, Vibration, Harshness' (NVH), design or usability. For example, a topic that benefits a lot from such an approach is the introduction of Digital Prototypes (DPs) (Breitling, Großmann, and Zöller 2009), a term that describes the shift from replaceable physical to virtual methods in the product development. The challenges the DPs have to overcome are congruent to a large part of possibilities that emerge from this thesis' challenges, such as an early assessment of innovations, a detection and handling of conflicting goals, an efficiency enhancement through the use of shared and structured knowledge, a shortening of the development time, a reduction of the development costs, an enhancement of the product quality, an efficient assessment of construction alternatives, a holistic and reliable assessment, optimal fulfillment and optimizing of all calculable vehicle properties analogues to physical prototypes and an increase of prototype maturity in early development stages (ibid.).

Although the trend in product development shifts to the use of more and more virtuality, some prototypes and measurements still have to be conducted physically today. On the one hand, internal challenges hinder the implementation of virtual methods: some fields in the development are too complex to be solvable virtually, the assessment of a hypothetical virtual outcome too vacuous or the realizable shift too expensive. On the other hand, external demands, like legal regulations, stipulate physical experiments and developments, e.g., in order to satisfy the requirements for selected crash tests.

However, not only the methods themselves, i.e., their procedures, are getting more complex, but their relationships to their connected concepts, such as the involved resources, roles, tools, processes and further linked concepts. The reasoning and inference needed to derive and hence to make a decision for a suitable selection for a task at hand is difficult, among other things, because of a plethora of existing methods which entails the unawareness of their availability which in turn leads to the use of just a partial quantity and unevenly distributed knowledge about them. Furthermore, the formalized knowledge about those methods and their contexts has to be valid, complete and consistent in order to reason correct inferences. Therefore, a benchmarking and selection based on adequate decision criteria, such as time, maturity, quality and cost, is required and the method application in the processes has to be assessed, made comparable and available. For instance, depending on a CAx method's execution date in the business pro-

cess, the execution and therefore the result of a method varies in quality, cost or execution length, because the input parameters, e.g., the applied models, vary in their maturity. Next to the aforementioned metrics, more complex analyses are necessary in order to identity method development and improvement needs (*cf.* Weigt 2008).
Because the number of methods and the context information is increasing over time, we need a system that is on the one hand able to handle the complex analyses that include thousands of individuals efficiently and on the other hand is still usable with minimal effort and structured clearly for the respective user.

1.2.2. Objectives

Based on the challenges introduced in the previous subsection, the following objectives have been derived:

Objective 1 (*Method Integration*)
Develop an integrated method meta model to harmonize method semantics and structure while preserving their original meaning. Furthermore, include the method context, e.g., a method's description, stakeholders, tools and processes, in order to control the complex method landscape.

This method meta model shall include a method definition that, among others, can be applied to the CAx domain in an automotive enterprise and thus consolidate their descriptions. Thereby, it has to consider varying meaning, depending on the stakeholder, different vocabularies and the applicability of the methods, e.g., stated in regulations. The goal is a declarative description of methods across the entire development division and not a guideline on how to develop methods or meta methods. The resulting meta model will act as the basis for the communication with other business divisions, partners, suppliers and group brands.
In order to realize such a meta model, apply appropriate tools and suitable modeling languages that respect the separation of code, domain and business knowledge, such as rules, is powerful enough to express the business logic, is computable in reasonable time and is flexible and dynamic enough to cope with the fast-changing circumstances.

Furthermore, demonstrate how to integrate established standards, existing agreements and compromises and further enterprise models, like product models, Controlled Vocabularys (CVs) and metrics, like maturity and quality attributes. Because these information exist in various forms, show how to integrate and reuse existing data sources and meta models.

Objective 2 (*Methodology*)
Apply or develop a fitting methodology for developing, mapping, executing and maintaining the integrated meta model and according rules.

Demonstrate how domain and business knowledge can be leveraged by gathering and formalizing it with ontological domain models, rules and queries in the industrial product development. Thereby, consider implicit and explicit method knowledge, such as operational knowledge known by DEs and documents describing method knowledge in natural language or structured form.
The formalized method knowledge, that now exists in a standardized way, needs to be authored, maintained and executed, i.e., showcase its user interaction. Therefore, apply a KM methodology that includes all the necessary procedures, beginning with KA, over knowledge authoring to knowledge execution, which includes the coupling of operational and business knowledge. For example, show how new methods can be integrated into the meta model and how existing internal and external knowledge can be reused.
Next to the formalization of slowly evolving, often static, method domain knowledge, support the formalization of (dynamic) shorter-lived rules, e.g., regarding regulations or organizational constraints and hence separate lifecycles of (conceptual) domain knowledge, rules and program code.

Objective 3 (*Enterprise Architecture and Method Meta Model Integration*)
Integrate the developed method meta model into an Enterprise Architecture, in order to benefit mutually from the additional knowledge available in an enterprise.

The main objective is the mapping of method knowledge with an EA in the product development to exploit relationships and dependencies between methods, processes, such as a PDP, IT, business objects and the enterprise strategy. By

keeping the KBs separated, the input of strategic, business and IT decisions on methods and vice versa shall be inferred, while the knowledge can be maintained independently. Therefore, design analyses that query and evaluate the integrated method ontology and rules while considering different stakeholders, artifacts and views.

Objective 4 (*Method Analysis*)

Perform analyses based on the developed meta models to support stakeholders in their decision making, for example, the method selection.

Based on the previous objectives support stakeholders in their method decision and selection by designing suitable analyses concerning methods, method combinations and the linked concepts, such as processes or resources. Besides, the assessment of these analyses will support the numerous stakeholders to monitor and hence control the method portfolio, to identify lacunae, redundancy, conflicts, increase quality, coverage, reusability and to promote a targeted development, consolidation, reuse or shutdown of methods. Existing low-quality, missing or expensive methods can be identified and compared to each other if applicable.
A prerequisite for these analyses is the implementation of suitable method metrics that are the foundation for more abstract queries and the method assessment. Thereby, consider the various users' requirements and views on the business and IT.

Objective 5 (*Evaluation*)

Evaluate the integration, meta models, methodology and analyses by implementing appropriate models and prototypes. The evaluation shall be conducted in the product development of a large enterprise.

The approaches that deal with the above listed objectives shall be evaluated with suitable stakeholders by developing prototypical applications and performing case studies following the presented methodology. The integration of methods shall be conducted along a PDP in real industry case studies.

1.2.3. Approaches

Integrating the heterogeneous Method Landscape using Semantic Web Technologies

For creating a concerted definition for our method meta model we first analyze, compare and demarcate state-of-the-art method and process definitions in suitable literature. We then consolidate and customize an abstract method definition that fits to the product development environment. Another source for various definitions will be the already existing sources in the company itself. Many departments have their own understanding, requirements and usages of methods. We will survey these understandings and consolidate a semantic description which allows the integration of numerous method definitions, like virtual and physical ones, and the method context, such as tools and processes.
For the semantic description of our methods, we create a meta model, implemented using ontologies. Other technologies, like modern database management systems in combination with, e.g., process engines, could achieve similar results, but Semantic Web Technologies (SWTs) are predestined for distributed, often-changing models, rules and data. Ontologies allow an easy adaption and reuse of existing models and knowledge, for example, because the modeled entities feature a unique identifier, in addition to a flexibility in the syntax of the different concepts and entities, i.e., the vocabulary. Besides, ontological reasoning allows checking the consistency of our KB and provides the ability to infer new knowledge. One paradigm when working with ontologies is the recommended limitation of the ontology's scope to one single domain, though. That means that one monolithic ontology which covers all possible knowledge areas is not suggested. However, that does not mean that the different KBs cannot be linked to each other. In fact, having mappings between or integrating different domain ontologies (DOs) into a global ontology is usually a wise and necessary procedure.
Already existing data from sundry systems can be integrated by using Semantic Web middleware, e.g., triple stores or BRMS, with integrated built-ins, that support connections with databases. Besides, modern Semantic Web (SW) middleware is highly scalable, which is a requirement for the computation of the interconnected knowledge, including the integrated resources in our method model, in reasonable time.
DOs can be developed by different stakeholders, each one representing a subjective view on the same domain. When integrating these different ontology

modules, accumulated information can be computed or tailored views can be displayed. In combination with other SWTs, like query languages and rules, that offer another type of expressiveness, we can create an architecture that strictly separates business, domain and IT logic, i.e., the source code of our application and the business knowledge, including the vocabulary, all the semantic relations between the modeled entities and their instances. Thus, the software as well as standardized and formalized business and domain knowledge can be reused, integrated and maintained independently. Consequently, DEs can add and maintain their knowledge themselves, which leads to a shorter information publishing process and thus more up-to-dateness. Furthermore, ontologies can be used as reference ontologies and CVs in order to organize the business knowledge which further improves communication and collaboration. Web Ontology Language (OWL), especially the extension Simple Knowledge Organization System (SKOS), supports the use of various labels, hence different vocabularies can be attached to the concepts, if required.
Another task for this thesis is the aforementioned integration of existing resources. We will examine mapping techniques that enable us to connect our ontological and rule-based model to already in place data, i.e., we demonstrate how to integrate standards and further ontologies in the case studies with different techniques, e.g., by using mapping rules, to further concrete and make the method context information more comprehensive. As another case study, we will integrate different CAx Bill of Materials (BOMs) using ontologies and rules. Each CAx technology represents the actual technical content and configuration of a car in a CA-specific BOM. This allows us to compare the knowledge coverage of different CAx disciplines and the detection and hence reduction of knowledge gaps. The DEs shall be able to modify the mapping rules and the data models independently from the prototype application, that will be implemented as well.
The method meta model is the foundation for linking already in place method, method context and enterprise information that are spread across various systems, semantics and formats. Furthermore, the method metrics and analyses will be based on the integrated meta models.

Methodology for Semantic Web Technology-based Knowledge Management

We realize our meta models, KBs, prototypical applications, operational and business rules by following an SWT-based approach. Using SWTs bears many ad-

vantages: on the one hand, it enables a clear separation of domain and business knowledge, that can be extended and modified by domain and business experts. On the other hand, this knowledge can be implemented and managed independently from the applications that process it. Additional benefits and advantages have already been elaborated in the previous sections, though. Besides, SWTs' ability to handle complex and extensive meta models and KBs in the automobile development, expressed by ontologies and rules, has been demonstrated (Syldatke, Chen, et al. 2007). For example, SWTs have been used to model, analyze and validate the data recorded during HiL tests for Electronic Control Unit (ECU) specification validation (ibid.). Another example is the modeling and analysis of matrices in the PDP, like the one presented in Figure 1.2, or the description and analysis of dependencies in multi-domain configurations, such as a CAx architecture (Syldatke, Lutz, et al. 2008).

Creating applications and (meta) models that build on SWTs exhibits some similarities with software design in general. However, many aspects, like the separation of business and domain knowledge, differ. Because establishing such a methodology by ourselves and from scratch would have been a daunting task, we participated in the EC-funded FP7 project ONTORULE which resulted in the ONTORULE methodology for combining ontologies and rules and to "enable users, from business executives over business analysts to IT developers, to interact in their own way with the part of a business application that is relevant to them" (de Sainte Marie, Escudero, and Rosina 2011). The project consortium included top research institutions and vendors of BRMSs and KB systems along with industrial partners, such as Audi, that have been the test beds for the methodology. Building on the endeavors of the project, this methodology has been adapted, applied, further customized and augmented for this dissertation, most notably the different phases KA, authoring and then executing and maintaining the KB, for creating our ontologies, rules and queries, either in the case studies or the main chapters. Supplementary, we have adopted and defined a set of stakeholders, their roles in various methodology phases and their requirements concerning the analyses and views in our considered domains.

Naturally, an enterprise already has an established IT landscape, e.g., databases or wikis, where they describe parts of their method knowledge. Additionally, a lot of insights and descriptions about proceedings are either not put down in writing at all or just as office memoranda, though. Therefore, we examine different ways to acquire and reuse existing data and describe its integration. Aside from

conducting KA with business people, we also examine techniques like applying Natural Language Processing (NLP) to create ontologies or to extract candidate BRs from given regulations or any other textual descriptions of the relevant domain, for instance, demonstrated by a case study about a lexicalized ontology to define business vocabulary in section 7.2.

Enterprise Architecture Integration

In the first objective we tackled the problem of creating an integrated SWT-based meta model, rules and queries for method information in product development. When we broaden our KB with even more enterprise information we are able to answer even more complex queries for numerous stakeholders and have a sound foundation for knowledge sharing. Therefore, we demonstrate how to integrate an EA meta model and consequently analyze it.
First of all, we create a meta model based on the EAF The Open Group Architecture Framework (TOGAF) (The Open Group 2011) and the ArchiMate specification (The Open Group 2013). We use these standards, because of their popularity and penetration of the market – according to Cameron and Mcmillan (2013), the absolute majority of companies use a hybrid framework approach. However, 82,2 % of the survey's participants use elements from TOGAF for their hybrid EA-specific approach. When using a primary EAF, TOGAF is the leading framework, as well.
We implement our EA hybrid meta model as an ontology and demonstrate how to map the considered concepts to the method ontology (Rosina and Bauer 2015). Furthermore, we show how to integrate further EA extensions, such as process and product extensions, and present supported business goals following from this approach. In the Property-Driven Development (PDD) case study from section 7.5, we showcase a prototypical application based on a customized business scenario, along with ontologies, rules and queries, that, among others, integrates and analyses process, method and IT concepts.
The fourth objective, the analysis of method knowledge, deals with the analysis of the integrated knowledge in general, along with method architecture artifacts, views and analyses based on method and EA concepts, such as roles, actors, software applications, IT infrastructure and business strategy.

Method Analysis

Using SWTs to express and integrate method information and EA elements is the foundation for lifting the domain and business knowledge to an abstraction layer that allows analyses and assessments in order to monitor, select, adapt, reuse, shutdown or develop new and existing design methods and their related concepts.
First, we collect and elaborate the requirements, focusing on the business analyses that help to select the appropriate method for a designated scenario. That means we create a catalog with questions, such as "Which role uses which method?", "How can the methods be compared?" or "Which is the best (considering quality, time, cost) method for a scenario?". We will also consider questions regarding the dependencies between IT systems and methods, and will connect the method information with the associated product information, e.g., CAD models, physical parts or tools as demonstrated in the PDD case study. When the organizational and system prerequisites are met, we implement a set of queries to analyze the coherences between methods and their environment. The connected process information allows stating when a method can be executed at the earliest, the latest and what the buffer times are, considering metrics, such as the method's set-up time and execution time or the availability of required resources. Besides, business capabilities, which are directly connected to the enterprise's strategic layer will benefit from this enrichment of knowledge. New KPIs can be defined and strategical goals, e.g., the goal to increase the use of virtual methods, can be traced.
This is realized by the implementation of ontologies, rules and associated queries with languages like Frame Logic (F-Logic), ObjectLogic, OWL and SPARQL. Afterwards, a demonstrator is evaluated in order to validate the previously gathered requirements.
Furthermore, we demonstrate how methods and tools can be selected, classified, analyzed and assessed in general. On the one hand, by matching method outputs and inputs, we can create method chains and their linked tool chains that support the realization of process actions. Method alliances, on the other hand, present alternatives for the realization of such a task. In order to propose the best suiting method combination for task, we consider metrics, such as time, quality and cost, and demonstrate how a metrics ontology can be implemented and matched to our method ontologies.

Furthermore, we define stakeholder roles, views based on these stakeholders and method architecture artifacts that represent conceptual method analysis results, such as catalogs, matrices and further diagrams, including queries regarding the data quality which support DEs, Knowledge Engineers (KEs) and managers to rise overall quality.

Summarized Approach

During the product development, numerous stakeholders are confronted with various *knowledge types*, that can be partitioned into *implementation* and *administration* knowledge categories, as depicted in Figure 1.4. "The *product knowledge* includes all knowledge about an existing or planned product" (Roth, Binz, and Watty 2010) and is a linkage between those two categories. Various *characteristic properties* can be connected to the regarded *knowledge types*, such as *tacit* or *explicit*, *structured* or *unstructured* and *present* or *future*. The assigned numbers for each *knowledge type* symbolize their respective significance within a particular PDP phase (ibid.).

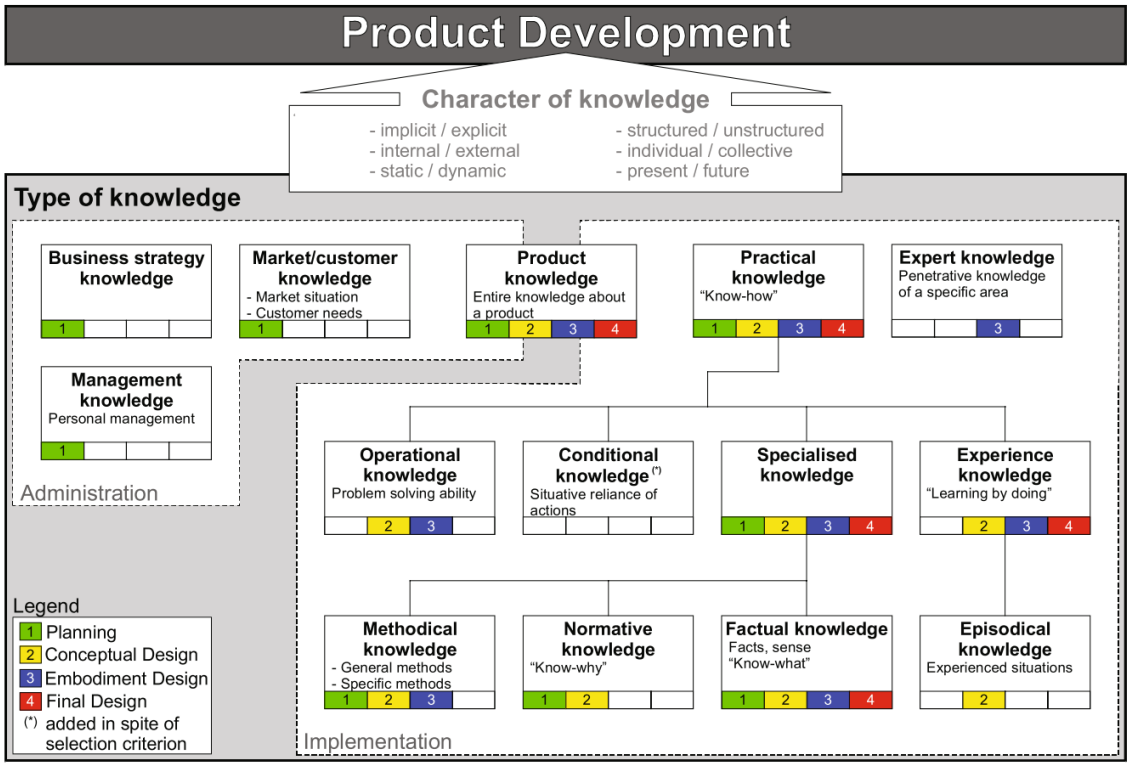


Figure 1.4.: A theoretical model of the structure of knowledge in the product development (Roth, Binz, and Watty 2010).

One aspect of this wide array of *knowledge types* is the *method knowledge* and the *method contexts* required to perform tasks in the PDP which can also be classified using the *characters of knowledge* presented in Figure 1.4, i.e., we consider different forms of knowledge, like documents, data bases or non-written knowledge. This knowledge can be used to describe a baseline architecture (*present*), a target architecture (*future*) and it might be *static* or *dynamic*.
Specialized and *factual knowledge* can be stored using ontology languages or other vocabulary standards, such as OWL resp. Semantics of Business Vocabulary and Business Rules (SBVR). Next to ontologies, the business knowledge can be formalized using rules and queries. *Normative knowledge* is tightly connected to a method's goal which can in turn be linked to an enterprise's strategical goal which is regarded in EAM. Likewise, we can match the remaining knowledge types roughly to the orchestrated KBs considered in this dissertation that have the purpose to support stakeholders by storing, inferring and providing information about areas, such as *methodical*, *expert*, *management*, *product* or *business strategy knowledge*. It is not possible to match unequivocally, though, because of the *knowledge types*' vagueness. Besides, some types of knowledge, like talent, skill or experience are often *tacit*, i.e., they can hardly be formalized.
This thesis showcases how various *knowledge types* in the product development can be acquired, formalized and integrated using SWTs which allows a concerted analysis and assessment of methods and the method contexts and dependencies, like processes, resources and elements, such as roles, tools and enterprise goals, that are partially modeled in an EA. As a consequence, the knowledge and logic will be separated using ontologies, rules and queries which allows the right roles to interact with their own knowledge in their preferred terminology, independently from the other KBs' lifecycles and responsibilities.
The case studies and evaluations are conducted in the product development of a large automobile manufacturer.

1.3. Publications

The published work concerning this thesis is presented in this section. Arising thereby, my own contribution in the relevant publications is clarified and related to the parts of this dissertation.

- Peter Rosina and Bernhard Bauer (2015). "Enterprise Methods Management using Semantic Web Technologies". In: *Fifth International Symposium on Business Modeling and Software Design (BMSD)*. ed. by Boris Shishkov. Milan, Italy: Scitepress. ISBN: 979-989-758-111-3

This paper primarily describes parts of chapter 6, i.e., the mapping between an EA and method ontology introduced in chapter 4 on a meta model level and is my own work.

- Melanie Langermeier, Thomas Driessen, et al. (2014). "Change and Version Management in Variability Models for Modular Ontologies". In: *16th International Conference on Enterprise Information Systems (ICEIS)*. Lisbon, Portugal
- Melanie Langermeier, Peter Rosina, et al. (2013). "Management of Variability in Modular Ontology Development". In: *Service-Oriented Computing - ICSOC 2013 Workshops*. Ed. by Alessio Lomuscio et al. Lecture Notes in Computer Science. Berlin, Germany: Springer, pp. 225–239

These two papers have been realized together with my colleagues at our chair. It deals with the mapping, merging, change and version management of ontology modules using feature models, covered in section 4.6. The described methodologies have been a product of our common efforts. My own work in these papers mainly dealt with the technical realization, related work and the foundations. However, content-related, we produced all the chapters together, so it is impossible to demarcate my own work on this level.

- Sonja Zillner et al. (2015). "Method of Composing an Integrated Ontology". US20150081648A1

This patent emerged from our work in the just now presented paper "Langermeier, Driessen, et al. (2014)", hence, my contribution is congruent to the explanation above.

- Christian de Sainte Marie, Miguel Iglesias Escudero, and Peter Rosina (2011). "The ONTORULE Project : Where Ontology Meets Business Rules". In: *Web Reasoning and Rule Systems* 6902, pp. 24–29

This paper introduces the ONTORULE approach at a mid-term stage of the project. My participation is mainly the evaluation, i.e., the presented case study about CAx method integration which is influential throughout the whole thesis.

- Nouha Omrane, Adeline Nazarenko, Peter Rosina, et al. (2011). "Lexicalized Ontology for a Business Rules Management Platform: An Automotive Use Case". In: *Rule - Based Modeling and Computing on the Semantic Web*. Ed. by Frank Olken, Monica Palmirani, and Davide Soltara. Vol. 7018. Springer Berlin Heidelberg, pp. 179–192. DOI: 10.1007/978-3-642-24908-2_21

The first phase of our methodology deals with KA, presented in section 2.5 and chapter 5. In this work, we have evaluated and introduced an NLP-based approach that allows extracting relevant text passages from written method and process descriptions, formalize them using SWTs and establish a traceability between source documents and ontological entities. My contribution has been the industrial case study description, the context introduction, and support during its implementation. This paper is revisited in section 7.2 as a case study.

The following publications are deliverables realized during the ONTORULE project and have been released in this context.

- Peter Rosina and Thomas Syldatke (2011). *D4.4 - Evaluation of the AUDI R&D Business Orchestration System*. ONTORULE Project

This document has been classified as confidential, because it contains a comparison with the company-internal requirements concerning the ONTORULE demonstrators that have been classified as confidential as well. Hence, our case studies in chapter 7 do not use these results and insights directly, but we mention them in a way more abstract and general way. However, the evaluation and the gathered experiences, feedback and insights have been very valuable, led to new and focused requirements and in turn influenced the whole thesis.

- Peter Rosina and Eva Maria Kiss (2011). *D4.3 - AUDI R&D Business Orchestration System*. ONTORULE Project

In this deliverable, we mainly present the initial PDD case study described explicitly in section 7.5, which is much more exhaustive, though. The modeling and technical realization of the SWT, the application and customization of the methodology and overall realization of the deliverable have been my own contribution. The initial business scenario behind the case study, *viz.*, the PDD, requirements, domain and business knowledge and logic, arose from DEs at Audi. A second, alternative technical realization of the case study using OWL RL and the CommonKADS Methodology (Schreiber, Akkermans, et al. 2000) has been realized by my co-author as described in the document.

- Peter Rosina and Thomas Syldatke (2010). *D4.2 - Semantic integration of BOMs - public demonstrator*. ONTORULE Project

This deliverable deals with the integration of different CAx BOMs using SWTs described in section 7.3. My contribution is similar to the previously mentioned deliverable, i.e., model, and implementation of the SWTs and realization of the document itself. Co-workers at Audi, i.e., DEs, contributed the business case and requirements behind this work.

- Eva Maria Kiss et al. (2010). *D8.4 - Market Intelligence Report*. ONTORULE Project

My contribution to this deliverable is confined to the case study concerning the CAx integration using technologies and approach of the project which is used throughout the thesis.

- Roman Korf, Eva Maria Kiss, Patrick Albert, et al. (2009). *D8.3 - Exploitation Plan*. ONTORULE Project

My contribution to this deliverable is confined to exploitation opportunities concerning the ONTORULE approach and our case studies, i.e., the use case descriptions, the possible impact, performed actions, IP protection/licensing and their relevance in general.

1.4. Outline

The following illustration in Figure 1.5 depicts the structure of the thesis, including the covered topics. Based on the table of contents, the figure is divided into the three main thesis parts along with the containing chapter titles seen in blue on the left side. The white elements on the right express the topics covered by the respective chapters if they are not self-explanatory by the chapters' titles.
In the first chapter of this dissertation, the thesis has been motivated and introduced, including the challenges to be tackled, the derived objectives and performed approach, followed by an overview of published work. The ensuing chapter 2 provides the conceptual and technical foundations for the main part, i.e., KM, SWTs (especially ontologies and rules), EAM, and the methodological approach. Furthermore, it delivers background explanations for the use cases

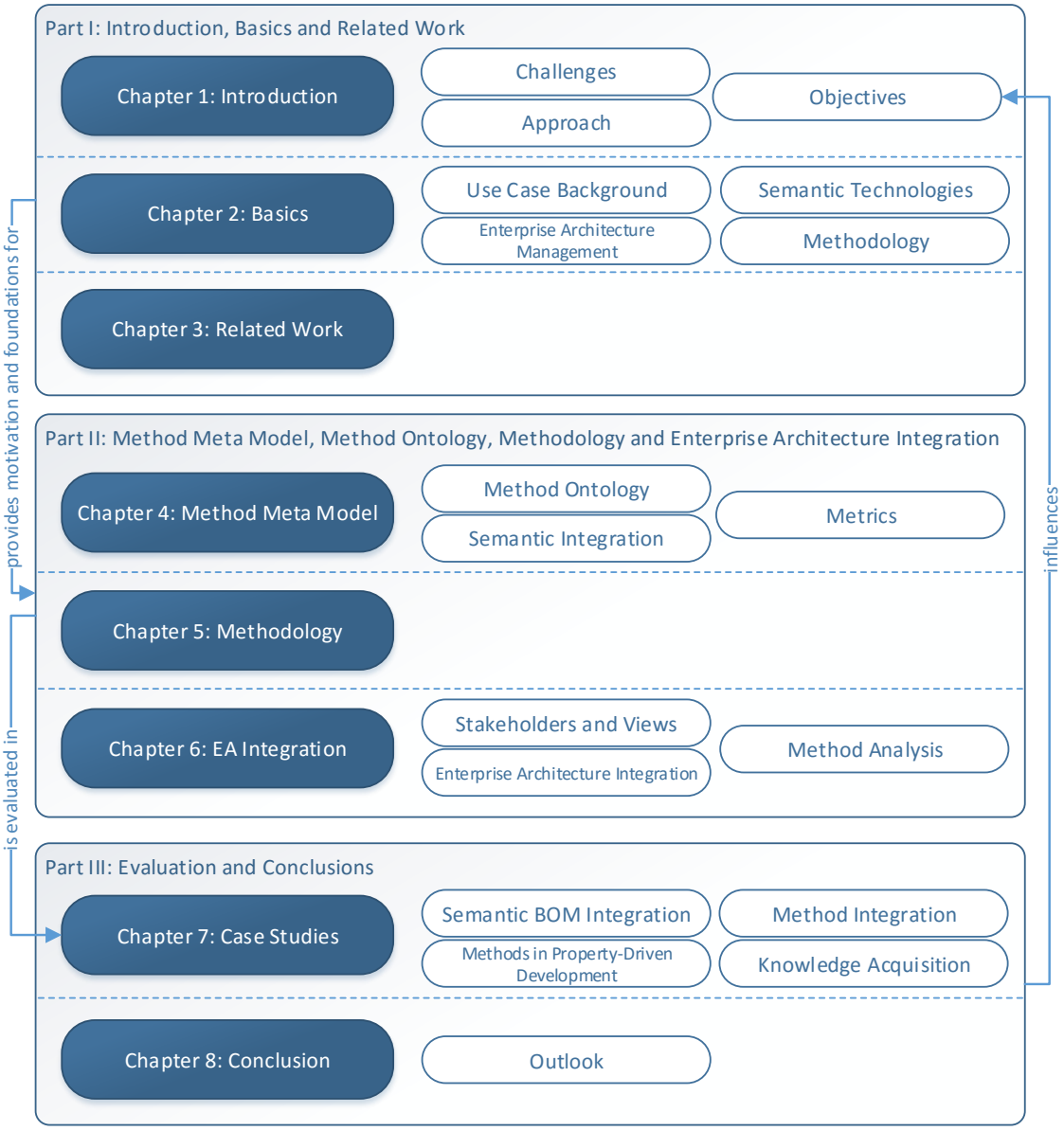


Figure 1.5.: Thesis structure.

and running examples that originate from the field of application of virtual product development. Part I of this dissertation is then completed by an overview about the related work (chapter 3) and a comparison with the approach concerning the succeeding principal part II.

The main part II is divided into three chapters: The first one introduces the method meta model in chapter 4, followed by the presented methodology in chapter 5 and finally chapter 6 about the method meta model's integration into an EA

meta model. Thereby, chapter 4 begins with a definition of the term *method* and its contexts, followed by the definition of a method ontology which realizes the meta model. Corresponding, the ontology's extension with further ontologies that cover the areas of metrics and KPIs, like a method's maturity or quality, is demonstrated. Next to this extension, the population of ontologies with existing data sources, the mapping and matching of ontologies by applying various strategies and the integration and linking of enterprise data using SWTs in general, are illustrated. Likewise, the fields of application that cover the use cases, i.e., resources, products, documents, etc., need to be integrated which is presented in section 4.6. An advantage of using ontologies is the viable enrichment of the modeled entities with various labels for the creation of a CV which is described as well. The section about related work in chapter 4 introduces existing method frameworks, metrics and modeling approaches. Afterwards, the chapter is concluded. In chapter 5, the approach for KM and SW application development is described. The up to here presented foundation and approach is then extended in chapter 6. The integration of the method ontology into an EA ontology, while respecting the various involved roles and hence required and customized views, is presented here. Furthermore, queries and models for method analysis and selection are introduced. Finally, the chapter is concluded as well.
The final part III, begins with the evaluation in chapter 7 that has been performed with various case studies that originate from real industrial use cases. Besides evaluating minor case studies, the main contributions demonstrate the semantic integration of data in the form of BOMs and of course an industrial case study located in the PDP that exploits the thesis' findings, i.e., the methodology, method ontologies, their integration, analyses and selection scenarios. These insights and outcomes of the evaluation have influenced our objectives in a subsequent iteration which in turn is reflected in the principal part II. Finally, chapter 8 concludes the thesis, summarizes the contributions and gives an outlook on the work.
Part IV, the annex, which is not illustrated in Figure 1.5, contains the utilized and consulted online and printed sources, lists all depicted resources, *viz.*, figures, tables and listings, and comprises additional material concerning the case studies, i.e., ontologies, rules, queries and User Interface (UI) overviews.

"In an economy where the only certainty is uncertainty, the one sure source of lasting competitive advantage is knowledge."

Ikujirō Nonaka

2

Basics

2.1. Knowledge Management

The knowledge of an enterprise comprises all the data and information that is explicitly embodied in its computer systems or on paper and the implicit and tacit knowledge possessed by its stakeholders, i.e., "the valuable and highly subjective insights and intuitions that are difficult to capture and share because people carry them in their heads" (Nonaka 2007). In order to help the enterprise keep track of the obtained knowledge and ongoing efforts, it is crucial to promote *Knowledge Management (KM)* about the business, the Information Technology (IT) and other supporting processes. Otherwise, a strategical orientation of the company and therefore the long-term competitiveness is impeded.

For already a long time, enterprise knowledge has been complex. Nevertheless, knowledge in the world of business and IT is still increasing in volume as well as complexity (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). Humans have long since been unable to comprehend the abstruse coherences in detail. Therefore, a need to gather and formalize the knowledge has arisen in order to be made computable.

KM is an organizational approach that includes many sub-disciplines, beginning with *Knowledge Acquisition (KA)* – the elicitation, gathering, analysis, modeling

and validation of knowledge which is required for *Knowledge Engineering* and KM in general (Epistemics 2003). Furthermore, KM deals with the course of action in order to implement knowledge-centric approaches in an organization and how to treat, use and maintain this knowledge in order to generate competitive advantages (Davenport 1994).

We want to focus on the *knowledge engineering* part of KM, which is "an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise" (Feigenbaum and McCorduck 1983). Besides acquiring and integrating knowledge, knowledge engineering comprises the automatic processing of knowledge, i.e., the combination of explicit knowledge, the generation of results and of course the explicit provision of implicit knowledge. Tacit knowledge can hardly be formalized, though.

Another aspect of knowledge engineering deals with the representation of knowledge. Depending on the user agent, different views on the knowledge, including the relevant content, have to be prepared and delivered.

The KA process is either performed directly with a human, the Knowledge Owner (KO), or from other arbitrary sources, e.g., from natural language in spoken or written form (Potter 2003), which is known as the discipline of *Natural Language Processing (NLP)*. Figure 2.1 illustrates some sample KA techniques, ranging from conceptual to process knowledge on the vertical axis and from explicit to tacit knowledge on the horizontal axis. KA techniques are widely used in Knowledge Engineering Methodologies (R. Studer, Benjamins, and Fensel 1998), such as, CommonKADS (Schreiber, Akkermans, et al. 2000) or the Conceptual (Enterprise) Modeling, which is based on CogNIAM (Nijssen et al. 2012). Knowledge Engineering Methodologies are also applied in the Semantic Web (SW), e.g., for the creation of ontologies (*cf.* section 2.3) with frameworks (Noy and McGuinness 2000), such as, Protégé (BMIR 2015). The roles performing this KA are typically known as *Knowledge Engineers (KEs)* – experts for machine-interpretable knowledge representation and translators of knowledge sources into these structures.

Such a knowledge representation structure requires conceptual foundations that are solidly based on logical reasoning and set theory (Sowa 1976; Sowa 2010). Besides, this knowledge has to be understandable by a large number of people (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). Typically, ontologies, further explained in section 2.3, are a popular choice to represent explicit and declarative knowledge, for example, is-a-relations or conceptual knowledge. However,

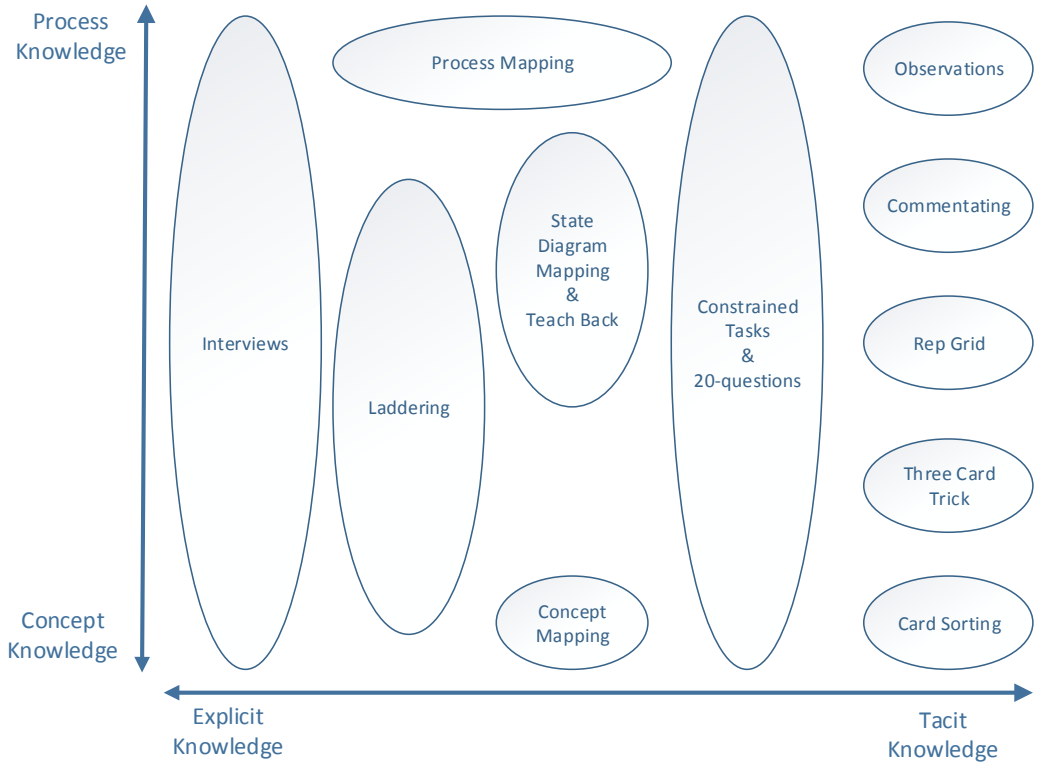


Figure 2.1.: Various KA techniques. Based on Epistemics (2003).

there are limitations for expressing procedural knowledge with ontologies. While it is possible to model processes, methods and workflows, this endeavor is wedded to some effort which is why KEs should aim to maximize the representations with explicit and declarative knowledge (Sowa 2010).
The structured knowledge, stored in a *Knowledge Base (KB)*, can then be computed and results be accessed by the end users.
Usually, this is done in an expert system – a software application that provides answers in lieu of human experts. The knowledge in such a system can be maintained and expanded by *Domain Experts (DEs)* and hence its capability to support decision making processes is improved. Expert systems are a type of decision support systems, have been a traditional application of artificial intelligence and there is still ongoing research in this field. They can be classified into their technological background, i.e., systems using case-based, model-based or rule-based reasoning and systems combining these approaches (Kiss et al. 2010).
Technologically, KM systems are evolving rapidly. Nowadays, there are various popular approaches for KM and knowledge learning, such as various forms of

machine learning, e.g., deep learning and big data approaches for application fields like NLP, image or speech recognition (Nielsen 2015). Other approaches utilize SW methods for KM (Decker 2002) or combine the previously mentioned techniques for creating sublime knowledge-based applications, such as IBM Watson[1] or Google Knowledge Vault (Dong et al. 2014).
Such systems, or more general, KM systems, are typically applied in enterprises or the World Wide Web (WWW) and exhibit one or more of the key characteristics (Probst, Raub, and Romhardt 2012), such as: knowledge goals, identification, acquisition, structuring, sharing, exploitation, preservation or knowledge assessment.

2.2. Semantic Web

Bringing the *"Semantic Web"* (SW) to fruition is an collaborative endeavor of various research groups, industrial partners and organizations pursuing the common goal to enrich the WWW with semantic data, thus transforming the web from an un- or semi-structured document-based technology to a knowledge-centered one. Berners-Lee et al. developed the vision of the SW in order to make the web's meaning machine-readable and -interpretable and hence integrate heterogeneous data and infer new information from existing KBs automatically (Berners-Lee, Hendler, and Lassila 2001). Thereby, the web contents will be annotated with semantic meta data. This effort is led by the World Wide Web Consortium (W3C) that supports developing and standardizes suitable technologies, knowledge representations and data formats including ontologies, rule and query languages. The W3C describes the endeavor as follows: "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries" (W3C 2013).
The prominent SW Stack in Figure 2.2 depicts the architecture of the SW. The lower layers provide the technological basement for the subsequent upper layers. The SW layer cake is still in development and is continuously improved. Layers containing formats, like Web Ontology Language (OWL) (W3C OWL Working Group 2012), Rule Interchange Format (RIF) (Kifer and Boley 2013) and SPARQL Protocol And RDF Query Language (SPARQL) (Harris and Seaborne 2013), are

[1] http://www.ibm.com/smarterplanet/us/en/ibmwatson/; visited on 08/18/2015.

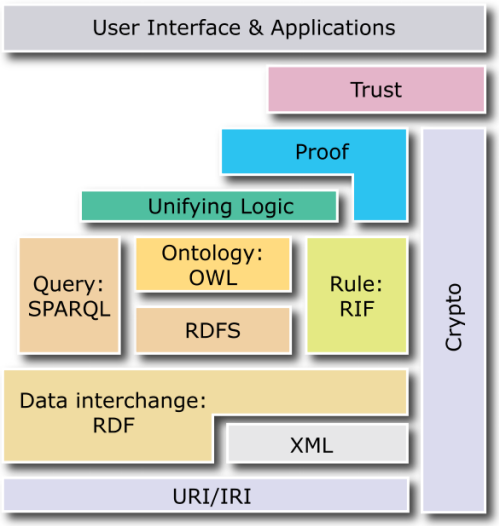


Figure 2.2.: SW layer cake (W3C 2007).

already standardized or currently in the process of standardization by the W3C. The remaining upper layers, i.e., the proof, crypto, trust and User Interface (UI) layers, are not yet realized. However, only the standardized entirety of all these layers realize the final vision of the SW.

The bottom layer realizes a direct link to the current web, using the well-known Uniform Resource Identifier (URI) standard that describes and thereby identifies the web documents. The Internationalized Resource Identifier (IRI) is a more general type of identification feature for resources that supports international characters. Hence, every resource has its own identifier. This layer combined with Extensible Markup Language (XML) is the basis for the Resource Description Framework (RDF) (Schreiber and Raimond 2014) which structures the resources in a graph-based data format (*cf.* section 2.3.3). Resource Description Framework Schema (RDFS) (Brickley and L. Miller 2014) uses RDF as its foundation but adds constructs to add more meaning to the data. Hence, it is possible to express class and property hierarchies, including domains and ranges of these properties (*cf.* section 2.3.3).

Even more functionality to express ontologies is added by OWL, which is described in detail in section 2.3.3. This knowledge representation language further extends RDFS, for instance, by making statements about the equality of ontological entities, class intersections, complements and unions or cardinality restrictions.

The standardized query language for RDF/OWL-based ontologies in the SW is SPARQL, which is comparable to SQL for databases. It does include communi-

cation protocols as well, but these are extraneous for this thesis. The rule layer, including the standard RIF, allows implementing production and logical rules. It was introduced because "users frequently encounter the need for specifying, triggering and executing arbitrary data transformations" (Janik, Scherp, and Staab 2011).
All the introduced and currently realized SW layers, especially ontologies, queries and rules, serve as the technological foundation for the following chapters. Next to the standards mentioned in this section, we apply further knowledge representation and rule languages, i.e., Frame Logic (F-Logic) and ObjectLogic, which are also presented in section 2.3.3. Moreover, we do not necessarily apply these technologies for the WWW, but for the enhancement and realization of enterprise domain models and their applications which is in line with Berners-Lee et al.'s SW vision, though.

2.3. Ontologies

Originally, the term *ontology* is rooted in philosophy where it represents the science of existence or being, a central branch of metaphysics.
In computer science, more precisely, in the branch of artificial intelligence, the term ontology stands for a technical artifact that is created to represent the knowledge of a specific domain. This knowledge can either be real or imaginary (Gruber 2008).
Ontologies are used to simplify the communication between machines and humans. They offer a (formal) way, to describe relations and concepts of a particular domain – a vocabulary.
On the one hand, they have a formal representation so they can be read, executed and written by computer programs. On the other hand, the coherences between the model elements are well readable by humans depending on the type of representation, for instance, in an ontology editor like Protégé (BMIR 2015). This can either be a graphical notation or a textual syntax, which makes the represented knowledge discussible and explicit.
Ontologies are part of the W3C as a standard for the SW. However, many ontology definitions with varying meanings exist (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). According to Dietz (2006), "the ontological model of the world consists of the specifications of its state space and its transition space".

2.3.1. Foundations

The Object Management Group (2009) defines an ontology as "the common terms and concepts (meaning) used to describe and represent an area of knowledge". That means, knowledge expressed in a taxonomy, a thesaurus, a conceptual model and even a logical theory can all be regarded as an ontology (ibid.), i.e., concepts that are somehow connected by relations is considered to be a sufficient model.
One of the most prominent definitions describes an ontology as a "formal, explicit specification of a shared conceptualization of a domain of interest" (Gruber 1993). That means that concepts of a particular knowledge domain can be related to each other by specified rules, interpretable by machines.
The modeled knowledge is often a trade-off between its application purpose and the "unique truth" (Parreiras et al. 2012). Making domain assumptions formal and explicit allows stakeholders to understand the vocabulary easier. Thus, ontologies are an instrument to discuss meaning and hence update legacy knowledge by making changes to revised domain assumptions (ibid.).
The term "*shared*" emphasizes one of the most important features of ontologies: it can be reused by others; persons and applications alike. This communicative feature can even lead to the standardization of specific ontologies that spread through the SW, which again leads to a consistent understanding of what information means.
In most cases, an ontology represents special knowledge of a specific domain, e.g., the domain of genes in biology or the domain of automotive parts. Other areas where ontologies are applied today are, for instance, NLP, medicine, pharmaceutical industry, enterprise integration, Enterprise Architecture Management (EAM), mechanical engineering, standardization of product knowledge, knowledge engineering, geographic information systems, legal information systems or electronic commerce (Guarino 1998). Such a specialized ontology is called *Domain ontology (DO)* in contrast to general *top level* or *upper ontologies*, like WordNet (Fellbaum 1998) or UMBEL (Giasson and Bergman 2015), that are not specialized in one domain (Hesse 2002), but describe "very general concepts like space, time, matter, object, event, action, etc." (Guarino 1998). Next to DOs, enterprises may also create *task ontologies* for modeling activities like selling or analyzing. Combining a domain and task ontology creates an *application ontology* as depicted in Figure 2.3.

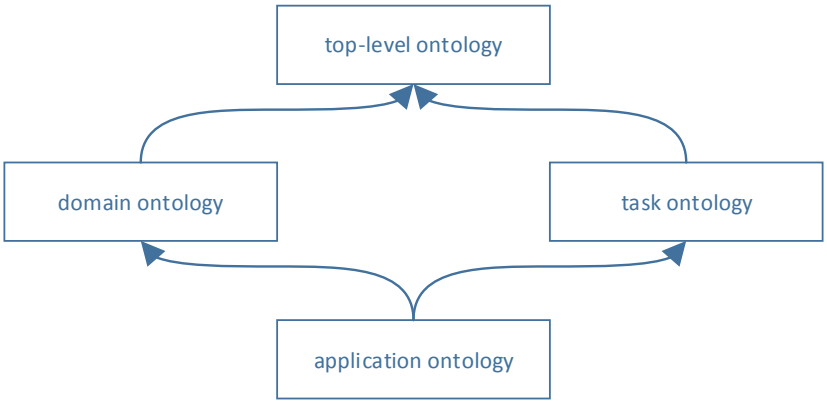


Figure 2.3.: Relations between kinds of ontologies. The arrows represent a generalization. Based on Guarino (1997).

The reasons for realizing an ontology are manifold. Noy and McGuinness (2000) enumerates the following ones: The *exchange of knowledge*, *reusability of domain knowledge*, *assumptions about the cohesiveness of concepts*, *separating domain knowledge and operational knowledge* and the *analysis of domain knowledge*. In the next paragraphs, we are going to explain these reasons and extend the list by some other important concepts, as well.

The *exchange of knowledge* about the structure of information between humans and software (Gruber 1993; Musen 1992) is the first reason. For example, if different web sites provide information in form of ontologies about its contents, software agents can analyze these ontologies and interrelate the provided information. These software agents would be able to answer user queries about the accumulated and interrelated knowledge and offer it in a new application. Obviously, a prerequisite for the exchange of ontological knowledge is a standardized knowledge representation format, like OWL (*cf.* section 2.3.3).

The *reusability of domain knowledge* is another important factor why an ontology should be developed. New ontologies can reuse existing ones for knowledge that is often needed. For example, the Friend of a Friend (FOAF) project offers the FOAF ontology that is devoted to linking people and information using the Web (Brickley and L. Miller 2014). Reusing existing DOs is also a time saver, because they do not have to be designed, created and tested again. Besides, ontologies that reuse existing DOs tend to be more compatible with other ones, because well-known terms and structures are considered and reutilized.

If *assumptions about the cohesiveness of concepts* are defined in an ontology — as opposed to classical software in combination with conventional relational databases —, lacunae, contradictory and incorrect knowledge can be replaced and added faster. Humans can also grasp the information about the structure more easily.
One of the most valuable features of an ontology is the capability to perform reasoning to infer new knowledge. *Reasoners*, software applications that process ontologies, are applied to infer new implicit knowledge from existing statements. Furthermore, they can verify the integrity and hence safeguard the correctness of the KB.
Next to storing an ontology on a file system, the KB can be stored in *triple* or *quad stores*, like 4store[2], OWLIM[3] or virtuoso[4], which are RDF-based databases that most often can process SPARQL 1.1 queries and support update mechanisms. *Quad stores* support the use of named graphs, i.e., the query can supply IRIs for the statement. The stores provide different reasoning and performance capabilities so the choice depends on the business case. The W3C member submission OWLlink (Liebig et al. 2011) is a protocol to communicate between semantic applications and a backend that implements reasoners like Pellet (Sirin et al. 2007) or FaCT++ (Tsarkov and Horrocks 2006).
By *separating domain knowledge and operational knowledge* the structure of a product, a process or a method with all its components can be recorded in an ontology. For example, a computer software can use this ontological knowledge to build a product, independently of the product's components (McGuinness and Wright 1998). This enables software to produce various outcomes by exchanging or modifying the underlying ontology, while the code remains unaffected. The separation also eases the maintenance, because, typically, domain, task and operational knowledge exhibit different lifecycles with varying responsibilities for each area of competence as previously depicted in Figure 1.3.
The formal *analysis of domain knowledge* in form of an ontology yields valuable information, if an existing ontology is reused or expanded (McGuinness, Fikes, et al. 2000). Furthermore, many ontology languages support the *annotation of meta information*. This way, information about authors, further descriptions, creation dates and places can be added. Another important feature is the support of labeling ontological entities in order to represent the same concepts syntactically differently, e.g., for multilingual uses.

[2] http://4store.org/
[3] http://www.ontotext.com/owlim/
[4] http://virtuoso.openlinksw.com/

2.3.2. Description Logics

"Description Logics (DLs) (Baader, Calvanese, et al. 2003; Baader and Sattler 2001; Calvanese et al. 2001) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. The name description logics is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL; on the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics" (Baader, Horrocks, and Sattler 2007).
Systems that are based on DLs are common in many domains, such as "conceptual modeling, information integration, query mechanisms, view maintenance, software management systems, planning systems, configuration systems, and natural language understanding" (Object Management Group 2009).
Typically, a DL-based KB consists of two different components. The "terminological component" or Terminological box (TBox) contains axioms that describe a set of definitions and specializations, e.g., it expresses concepts and concept hierarchies or in other words classes, roles and the relations between them. This component corresponds to the schema of a database. The "assertional component" or Assertional box (ABox), on the other hand, contains assertions about the individuals in the domain which is comparable with the data inside a database. Assertions can also be considered as facts, for instance, OWL is restricted to unary and binary facts. When talking about the entities that are modeled in a KB, "we shall refer to the part of a real world [...] as its Universe of Discourse (UofD)" (Baader, Calvanese, et al. 2003). In some definitions, ontologies can even contain an RBox, which is an extension to an ontology by a rule set.
When mapping the KB to MOF (Object Management Group 2011), instances correspond to *M1* and the terminological component corresponds to *M2* (*cf.* Figure 2.4). However, we have to keep in mind that OWL Full permits classes to be instances of other classes whereas different modeling languages like Unified Modeling Language (UML) have a strict separation of the meta-model levels (Object Management Group 2009). *M3* represents the meta meta modeling layer, i.e., the meta model to the DL used to express the KB, e.g., the OWL meta model. Note, that not all elements defined in OWL resp. RDF are mapped to *M2* but some

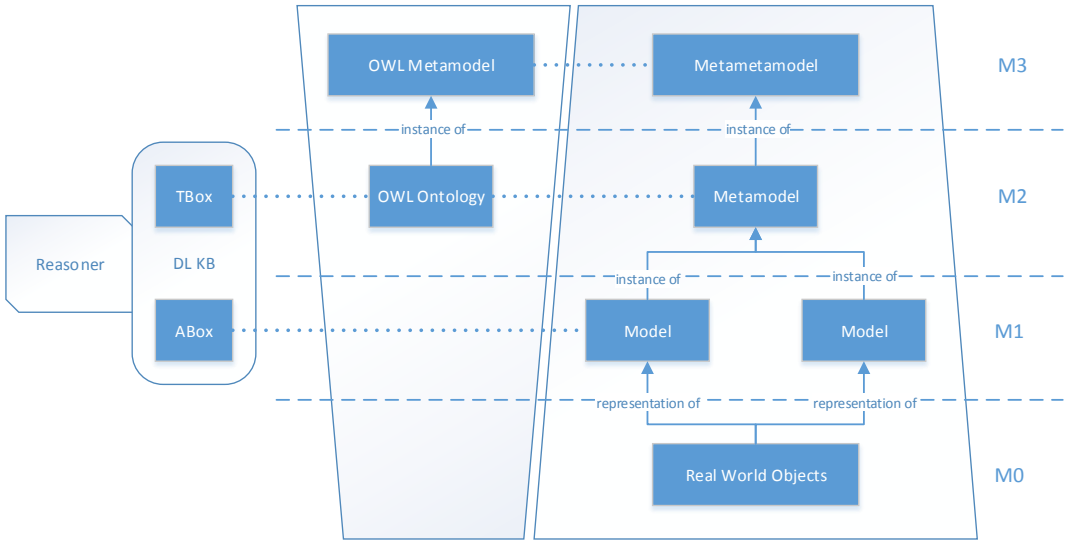


Figure 2.4.: ABox and TBox compared to the Meta Object Facility (MOF). Based on Staab, Walter, et al. (2010).

special cases, like data types (*String*, *Boolean*, *Float* etc.) or `owl:Thing` are mapped to *M1* (ibid.). Furthermore, Figure 2.4 depicts an integration of MOF and OWL, based on an *M3* bridge. It is also possible, though, to bridge models and ontologies on layer *M1* (Staab, Walter, et al. 2010), which means that the TBox is considered to be on the modeling layer *M1*.

DL appears in many languages, dialects and flavors. The most important DL is called $\mathcal{ALC}$ ($\mathcal{A}$ttributive concept $\mathcal{L}$anguage with $\mathcal{C}$omplements). It includes the most widely used constructs (conjunction, disjunction, negation, existential restriction and value restriction) in DLs (Baader, Horrocks, and Sattler 2007). One of the most prominent ontology languages is OWL which "is a member of the [DL] family of knowledge representation languages" (Object Management Group 2009). However, "among the myriads of the many DL languages, the most prominent one is the $\mathcal{SH}$ family" (Horrocks, Sattler, and Tobies 2000). The most widely adapted version of OWL is called OWL DL and uses the DL expressiveness $\mathcal{SHOIN}(\mathcal{D})$ while the newer (and now standard) language OWL 2 DL provides the expressiveness of $\mathcal{SROIQ}(\mathcal{D})$. The symbol $\mathcal{S}$ in these languages' expressivenesses is an abbreviation for the already mentioned $\mathcal{ALC}$, extended by transitive roles ($\mathcal{ALC}_{R+}$). The other symbols stand for, e.g., $\mathcal{I}$nverse properties and the $\mathcal{O}$ for nominals, which allows using individual names in concept descriptions (Baader, Horrocks, and Sattler 2007). For more information on DL naming conventions see "Description Logic Terminology" in Baader, Calvanese, et al. (2003).

Most of the DLs (with some exceptions) can be expressed in traditional First-Order Logic (FOL) which deals with "quantification, negation, and logical relations as expressed in propositions that are strictly true or false" (Object Management Group 2009). Traditional FOL also "specifically excludes reasoning over relations and excludes using the same name as both an individual name and a relation name" (ibid.). However, the main reason why DL is used instead of FOL to represent KBs is its decidability in finite time. Decidability means, that the reasoning performed with such DLs is effective, i.e., it terminates with a result that states if a given statement does or does not follow from given axioms.
Furthermore, OWL DL is also complete, which means that it "exhibits *complete* knowledge if and only if for every sentence α (within its vocabulary), either α or $\neg\alpha$ is known" (Brachman and Levesque 2004).
By reasoning over an ontology, we can also check whether it is consistent: "[...] a KB exhibits *consistent* knowledge if and only if there is no sentence α such that both α and $\neg\alpha$ are known" (ibid.). Patel-Schneider et al. (2004) define consistency as a "collection of abstract OWL ontologies and axioms and facts [which] is consistent with respect to datatype map D iff there is some interpretation I with respect to D such that I satisfies each ontology and axiom and fact in the collection". That means, that an ontology O is consistent (or satisfiable) w.r.t. to a datatype map D if a model of O w.r.t. D and and the vocabulary elements V exists (Motik, Patel-Schneider, et al. 2009).
Most modern reasoners, such as FaCT++, RACER, DLP and Pellet, implement the tableau reasoner which uses the analytic tableau method. The reasoning over a KB in DLs typically supports a set of standard inferences and algorithms:

- Subsumption checking: returns whether every instance of concept c is also an instance of concept c'.
- Instance Algorithm: whether or not some concept c is subsumed by another concept c' (Brachman and Levesque 2004).
- Satisfiability: whether or not some instance i satisfies a certain concept c (ibid.).
- Classification: finds all concepts c, a given instance i belongs to.
- Instance retrieval: finds all the instances of a given concepts or role.
- Query answering: a special kind of instance retrieval.
- Justification: When a reasoning result a in ontology O is given. Which is the minimal subset of the axioms in O that are responsible for this result?

Furthermore, ontologies can be distinguished by the Closed World Assumption (CWA) and the Open World Assumption (OWA). The first one implies that everything which is unknown is false, i.e., "unless an atomic sentence is known to be true, it can be assumed to be false" (ibid.). This inference rule in Logic Programming (LP) is called Negation as failure (NAF).
OWA, on the other hand, states that everything which is unknown is undefined.

2.3.3. Ontology Languages and Standards

As explained in the prior section 2.2 about the SW, the foundation of OWL is based on RDFS and RDF. Next to these OWA-based languages, we want to introduce the CWA-based ontology languages F-Logic and ObjectLogic, because parts of this thesis have been implemented and evaluated using these ontology languages. The following table 2.1 depicts the terms used in combination with the aforementioned ontology languages. Depending on the situation, we interchangeably use these terms for the considered meta model elements.

FOL	**DL**	**RDF/OWL**	**F-Logic**
Class	Concept	Class	Class
Property or Predicate	Role	Property	Relation
Object	Individual	Individual	Instance

Table 2.1.: Synonyms.

Resource Description Framework (RDF)

The Resource Description Framework (RDF) is the standard model for representing and interchanging information about resources on the Web, defined by the W3C (Schreiber and Raimond 2014). The framework's abstract syntax represents a direct, labeled graph. It is based on triples, that connect the modeled resources by a statement in the form of subject–predicate–object as depicted in Figure 2.5. These resources, e.g., objects, people or abstract concepts, are identified by IRIs which is a generalization of the URIs that allows non-ASCII characters. RDF can be serialized in many formats, for instance, Turtle (Prud'hommeaux and

Figure 2.5.: An RDF graph depicting two nodes connected by a triple.

Carothers 2014), JSON-LD (Sporny et al. 2014), Resource Description Framework in Attributes (RDFa) (Herman et al. 2015), N-Triples (Beckett 2014) and RDF/XML (Gandon and Schreiber 2014).

Resource Description Framework Schema (RDFS)

The Resource Description Framework Schema (RDFS) (Brickley and L. Miller 2014) introduces a vocabulary and semantics for structuring RDF resources by providing means to group resources (classes) and to enable relationships between them (properties). This system "is similar to the type systems of object-oriented programming languages such as Java" (Object Management Group 2009). They also exhibit some important differences, though. In object-oriented programming languages, a class is defined by the properties its instances may have. Here, it is the other way around: properties are described by "the classes of resource to which they apply" (ibid.).

```
<Person> <type> <Class>
<is a friend of> <type> <Property>
<is a friend of> <domain> <Person>
<is a friend of> <range> <Person>
<is a good friend of> <subPropertyOf> <is a friend of>
```

Listing 2.1: "An informal example of RDFS triples" (Schreiber and Raimond 2014).

The introduced classes and properties can be ordered in a hierarchy and instances are related to a class by the type property. Another important features of RDFS is the ability to define a property's domain and range, i.e., type restrictions for the subjects and objects of a particular triple as exemplified in Listing 2.1.

Web Ontology Language (OWL)

The most prominent and most widely adapted language to describe ontologies is the "Web Ontology Language" (OWL). It is recommended by the W3C for the SW (see section 2.2) and is the de facto standard in today's ontology applications. In comparison with RDF and RDFS it adds several features, extending these languages "by providing additional vocabulary along with formal semantics" (McGuinness and F. v. Harmelen 2004). These are, for instance, "class intersection, union and complement, local property restrictions, cardinality restrictions, and reflexive, symmetric, functional, transitive and inverse properties" (ONTORULE 2011).

OWL is the successor of the superseded language DARPA Agent Markup Language (DAML)+Ontology Inference Layer (OIL), but is also influenced by DL in general, from the frames paradigm and from RDF (Horrocks, Patel-Schneider, F. V. Harmelen, et al. 2003). DAML is a program of the US Defense Advanced Research Projects Agency (DARPA) which had the goal to make the contents of the WWW machine-readable. Later, they cooperated with additional institutes and the successor DAML+OIL emerged in the year 2000.

The version 1 of OWL (Bechhofer et al. 2004) was recommended by W3C in 2004; the syntax of the structural layout was based on XML. It was divided in three different sub-languages: OWL-Lite was designed for users that wanted to create simple ontologies, like basic taxonomies. It offered only limited expressiveness, e.g., cardinalities could only attain the values 0 or 1 and it did not allow the use of nominals. OWL-Lite is a subset of OWL-DL, hence, everything that can be modeled in Lite is also expressible in DL. In addition to that, OWL-DL, which is based on the DL expressiveness $\mathcal{SHOIN}(\mathcal{D})$, guarantees decidability (the reasoning finishes in finite time) and completeness (every possible inference is computable) of the applied semantics. In order to guarantee these characteristics, some modeling restrictions in the RDF structure are to be satisfied, e.g., a class must not be an instance of another class. If the KE does not want to be constrained at all when modeling in OWL, the third version, OWL-Full has been designed. In this case, however, the KE has to keep in mind that OWL-DL guarantees do not hold anymore.

Since 2009, OWL 2 has become the first version's successor and an official W3C recommendation (W3C OWL Working Group 2012). It is based on the DL expressiveness $\mathcal{SROIQ}(\mathcal{D})$ and features new things, such as property chains, punning,

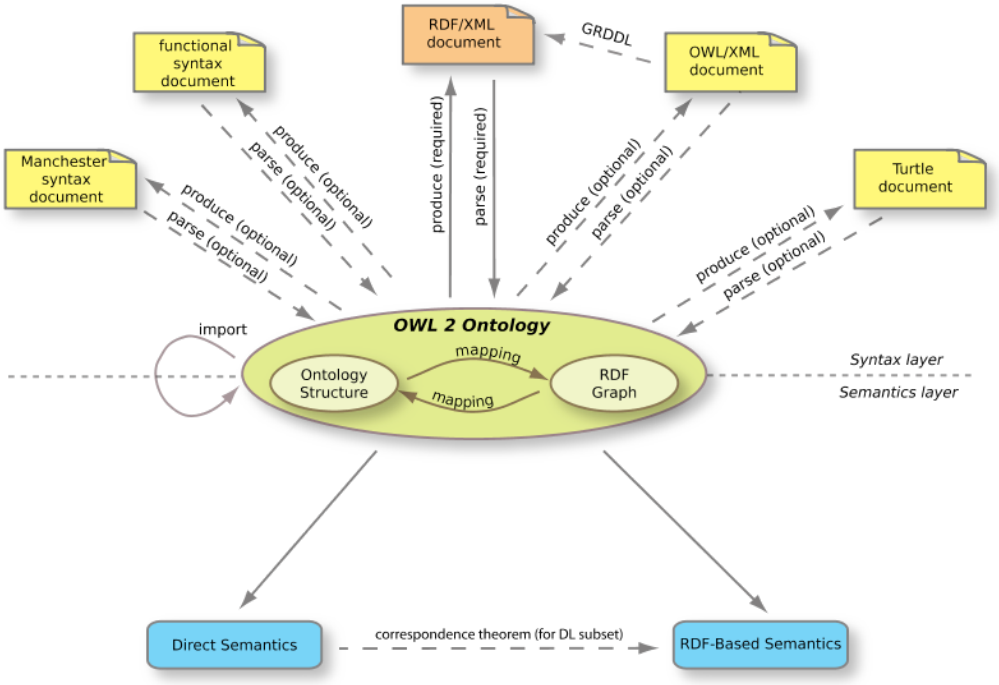


Figure 2.6.: The structure of OWL 2 (W3C OWL Working Group 2012).

keys, qualified cardinality restrictions and some syntactic sugar. It is still possible to write arbitrary, but undecidable, RDF graphs using OWL 2 Full but in contrast to OWL, it also features three OWL 2 DL language profiles, that are optimized for their specific field of application.

OWL 2 EL is meant for ontologies with a huge TBox, performing in PTIME-Complete. OWL 2 QL is a language profile meant for querying large data sets efficiently, i.e., with a runtime of LOGSPACE (more precisely, in AC^0, the same as relational databases performance) (W3C OWL Working Group 2012). The third profile, OWL 2 RL, "is amenable to rule processing; i.e., a standard forward chaining rule reasoner can be used for reasoning with OWL 2 RL ontologies" (ONTORULE 2011), also performing in PTIME-Complete.

Nowadays, a lot of notations have been developed in addition to RDF/XML (Gandon and Schreiber 2014), e.g., the Manchester OWL syntax (Horridge et al. 2006), Notation3 (N3) (Berners-Lee and Connolly 2011) and the notations listed in the previous section 2.3.3 about RDF. Turtle is subset of N3 and is a simplified and "a compact, non-XML based serialization of RDF" (Wood 2010). The different syntactical representations are depicted in Figure 2.6 on the syntax layer in addition to the semantics layer.

In this thesis, OWL has been used in many projects, e.g., for applying NLP methods or designing DOs for the case studies. These OWL ontologies, however, have been transformed into F-Logic for the integration of the Bill of Materials (BOMs) (*cf.* section 7.3) and transformed into ObjectLogic for the Property-Driven Development (PDD) case study (*cf.* section 7.5).

Frame Logic (F-Logic) and ObjectLogic

"Frame Logic" (F-Logic) (Kifer, Lausen, and Wu 1995) is a deductive and object-oriented LP ontology and knowledge representation language. Instances are part of classes and the relations between those objects are represented by methods (Traczyk 2005). In contrast to OWL, F-Logic uses the inference rule NAF, i.e., it implies a CWA. However, it is possible to combine OWL and F-Logic KBs (Kattenstroth, May, and Schenk 2007) and use them in parallel as shown in the case studies. Nevertheless, combining OWA and CWA approaches in general is difficult and can lead to undecidability (*cf.* section 2.4).
When implementing a KB with F-Logic, the language can be split into different sections, namely, the schema, facts, rules and queries; exemplified in the following Listings, which are inspired by various sources (Angele, Kifer, and Lausen 2009; FORUM Working Group 2008; ontoprise GmbH 2007).
In the first example, depicted in Listing 2.2, we can see a schema with the definition of subclasses (`Man` and `Woman` are `Persons`). The relation between two `Persons` is `marriedTo`, which features the attribute *symmetric* and a cardinality restriction.

```
Man::Person.
Woman::Person.
Person[marriedTo{1:1,symmetric}=>Person].
Person[hasSon=>Man].
```

Listing 2.2: F-Logic Schema.

In the next Listing 2.3, we can see the facts, i.e., a collection of the objects with their corresponding classes and methods. Instead of using F-atoms, as seen in the example, we could also use predicate symbols (P-atom), like $marriedTo(Alice, Bob)$.

However, we will mostly use F-atoms in the case studies, because it provides a better readability and separation between ontology and rules.

```
Alice:Woman.
Bob:Man.
Alice[marriedTo->Bob].
```

Listing 2.3: F-Logic Facts.

Rules consist of a rule head and body or *premise* and *conclusion*, explained in more detail in the following section 2.4. They are used to derive new information from given facts by LP.

```
FORALL ?X,?Y ?Y:OldPerson <- ?Y:Person[hasAge->?X] and ?X > 70.
FORALL ?X,?Y ?X[hasSon->?Y] <- ?Y:Man[hasMother->?X].
```

Listing 2.4: F-Logic Rules.

A query is like a rule with an empty *premise*. The final example in Listing 2.5 depicts a query that would return all objects that are women and their sons, having a father who is Bob.

```
FORALL ?X,?Y <- ?X:Woman[hasSon->?Y[hasFather->Bob]].
```

Listing 2.5: F-Logic Queries.

Next to the basic statements shown in the above examples, F-Logic offers a bunch of useful functionalities, like built-in features, for instance, string handling and aggregations, and features such as data base access, lists, namespaces and module support (ontoprise GmbH 2007).
Since 2010, F-Logic has been superseded by the ontology, rule and query language *ObjectLogic*, which is "best of breed of RDF, OWL and F-Logic concerning the expressive power and evaluation performance" (semafora systems 2012). Both languages are supported by the reasoner KAON2, which uses algorithms to reduce a $\mathcal{SHIQ}(\mathcal{D})$ KB to a disjunctive datalog program (Motik 2015).

2.4. Rules

In the SW Stack, ontological and query knowledge can be supplemented with otherwise inexpressible knowledge deposited in rules. Basically, a rule is a statement that includes a *premise* (or *precondition/precedent*) and a *conclusion* (or *postcondition/antecedent*), whereas the latter must be valid, when the *premise* is satisfied (Hitzler, Krötzsch, and Rudolph 2009). That means, any sentence of the form "`if [...] then [...]`" can be regarded as a rule (ibid.).
Rules can be formalized using various rule languages, for example, *Semantic Web Rule Language (SWRL)* (Horrocks, Patel-Schneider, Boley, et al. 2004), *F-Logic* (Kifer, Lausen, and Wu 1995) or *DL Rules* and *DL-Safe Rules* (*cf.* Hitzler, Krötzsch, and Rudolph 2009).
Besides, any programming language can obviously express rules. In this dissertation, we concentrate on rules that are expressed in "*Logic Programming*" (LP) and rules that are called *business rules (BRs)* or *production rules*, i.e., *Event Condition Action Rules*. Production rules resemble program statements, i.e., they are *operational*, and are usually executed actively (ibid.), commonly in a *Business Rule Management System (BRMS)*.

Definition 2.1 (*Business Rule*)

> *"A [BR] is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business" (Business Rules Group 2000).*

In our context, the following definition is even more helpful, as it has been used in the context of ONTORULE and defines a rule, not only BRs:

Definition 2.2 (*Rule*)

> *"[...] a rule is a statement expressing a validation or derivation rule. Depending on the context, validation rules (also called constraints) may be seen as part of the ontology, which leaves derivation rules (also called decision rules) as a more restricted definition" (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012).*

Validation rules are also called integrity rules (ISO 1987). Another meaning for a rule is an instruction for a person, or a combination of the before-mentioned definitions.

Moreover, query languages like SPARQL can be regarded as rules – they can extract information from a given KB and then extend the existing data with new conclusions (Hitzler, Krötzsch, and Rudolph 2009), for instance, by making use of SPARQL's *CONSTRUCT* statement. Hence, a query can be considered as an antecedent of a rule.
Formalized BRs are heavily used in organizations, where many rules and policies apply, for example, in banks or insurance companies. However, all companies have BRs, even if the company uses them unwittingly. They apply to either customers, employees, suppliers, processes or other agents, like computer systems. For example, a BR can state that a returning customer can be granted a discount of 5 % on his next purchase. More general, BRs "can specify who can do what under specified conditions, the combination of inputs and controls needed, and the resulting outputs" (The Open Group 2011). These rules do not necessarily have to be written down formally and certainly not in a *Business Rules Language (BRL)*. Nevertheless, following the BRs approach, i.e., creating a formal specification of BRs, for instance, managed and executed in a BRMS, can support companies to discover complex decisions, validate forms and data with constraints or to comply with policies and regulations. Therefore, axioms specified in an OWL ontology can also be regarded as a rule, since they feature a premise and conclusion, such as the statement in Listing 2.6, expressing that if a person is the author of a book then this person belongs to the class `Bookauthor`.

$$Person \sqcap \exists authorOf.Book \sqsubseteq Bookauthor.$$

Listing 2.6: An example rule in OWL DL (Hitzler, Krötzsch, and Rudolph 2009).

With the publication of OWL 2 in 2009, even more statements in DL could be realized, like expressing an uncle relationship between two entities. However, many statements are not expressible in DL, but in a Domain Specific Language (DSL) for rules, such as calculations and aggregations, e.g., *sum*, *max* or *avg* or, for instance, a rule that is defining a class `C` of children whose parents are married as seen in Listing 2.7.

$$hasParent(x,y) \wedge hasParent(x,z) \wedge married(y,z) \rightarrow C(x)$$

Listing 2.7: Example rule that is not expressible in DL (Krötzsch et al. 2011).

Approaches that combine DL languages, like OWL, and other formalisms, like F-Logic, an LP language, feature many advantages (Kattenstroth, May, and Schenk 2007; Krötzsch et al. 2011).
One appropriate use case, for example, is the ontology and data integration (Kattenstroth, May, and Schenk 2007).
Combining rule languages and ontologies in general is a current research topic and features many challenges. For example, the integration of OWA from the DL theories and the CWA from rule languages is very difficult, because it can lead to undecidability. An overview of the state-of-the-art issues is given in (De Bruijn et al. 2009). Most notably, the new W3C standard RIF (Kifer and Boley 2013), which provides a basic logic dialect (Boley and Kifer 2013) and a production rule dialect (de Sainte Marie, Hallmark, and Paschke 2013), is a promising approach.
Using rules and formalizing them using a rule language can be very beneficial for organizations (Berners-Lee 2006; Gougeon 2003):

- Separating business knowledge and logic from the remainder, for instance, program source, which empowers the business user to express, manage and automate the business' driving knowledge. Thus, changes can be performed faster and easier than in an application that incorporates the business logic in the program source.
- They are better understandable than source code by business users, i.e., more accessible.
- The formalized representation is computable, i.e., inference and constraints can be applied. For example, rules can be applied in real time in a rule-based application, for instance, an e-commerce shop system, as exemplified in Listing 2.8.
- IT people get clearer and better quality requirements which leads to less inconsistencies and a better productivity.
- Rule-based systems are interoperable.
- They extend the expressiveness of shared knowledge.
- They interlock with other SW languages, such as RDF, OWL and SPARQL.
- "Serendipitous re-use of knowledge in Rule form" (Berners-Lee 2006).

```
IF (object.value ≤ userinput.max_value) THEN ↙
   ↳ addtoresults(object).
```

Listing 2.8: Example rule of an e-commerce shop system.

In order to manage BRs in an enterprise efficiently and correctly, the Business Rules Manifesto has been designed and published (Business Rules Group 2003). This document concisely explains the nature of rules, their concerns and how to create and apply them properly.

2.4.1. Rule Languages and Standards

In addition to the DSLs mentioned in this section's introduction, we briefly present the other languages mentioned and applied in the course of this thesis. *Object-Logic* and *F-Logic* have already been explained in the former section 2.3.3.
RIF (Kifer and Boley 2013) has been presented as part of the SW Stack. Formerly planned as an interchange format between rule systems and to facilitate rule set synthesis and integration, it is considered a complete rule language today, featuring various dialects for specific use cases.
SWRL (Horrocks, Patel-Schneider, Boley, et al. 2004) is enumerated in this section, because it is a prominent language based on *datalog* which is based on *FOL*, just as F-Logic. These languages combine LP and DL. However, since RIF is the new W3C recommendation and SWRL has the drawback of not being decidable without constraints (Krötzsch et al. 2011), we omitted its use.
As mentioned, the boundaries between query and rule languages are blurred. In particular, *SPARQL Inferencing Notation (SPIN)* (Knublauch, Hendler, and Idehen 2011) is a language that integrates *SPARQL* rules with OWL. SPIN has been applied in the course of this thesis for the prototypical modeling and integration of a method ontology (*cf.* chapter 4) and a Product Development Process (PDP) ontology.
Furthermore, during the ONTORULE project, we realized a particular use case with the WebSphere IBM/*ILOG JRules* BRMS[5] together with our partners.
The final important standard for this thesis is *Semantics of Business Vocabulary and Business Rules (SBVR)* (Object Management Group 2015). SBVR/Structured English (SE), a textual SBVR representation for the English language, has been applied to model and validate parts of our case studies. For the future, upcoming Object Management Group (OMG) standards like *Decision Model And Notation (DMN)* (Object Management Group 2014) or *Semantic Information Modeling for*

[5] http://www-01.ibm.com/software/integration/business-rule-management/jrules-family/; retrieved 07/28/2015

Federation (SIMF) (Casanave 2012) offer promising approaches. Alternatively to rule languages, business logic can also be formalized with decision trees, tables or other models. Besides, rules can be expressed in General Purpose Languages (GPLs) and DSLs from other domains, like *UML*, *Business Process Execution Language (BPEL)* or *Business Process Model and Notation (BPMN)*.

2.4.2. Rule Engines and Management Software

As mentioned, formalized BRs are commonly executed in a rule engine, often part of a BRMS which is a "piece of software that supports [and manages] a large part of the [BRs] lifecycle within an organization" (Berrueta et al. 2011). Typical popular representatives of rule engines are, for instance, Drools[6], Jess[7] and WebSphere ILOG JRules BRMS[8].
A major part of production rule engines rely on the evaluation strategy *Rete* (Forgy 1982). This algorithm performs forward and, in more recent versions, backward chaining in order to infer the knowledge. Nevertheless, several other, in some cases more modern, rule algorithms are applied. Besides, rule engines typically can perform side-effects on the runtime environment, e.g., modify the *working memory* (*cf.* Brachman and Levesque 2004, chap. 7), print values or execute other arbitrary system commands, depending on their access rights.
However, in this dissertation, we also apply LP languages that exhibit rules.
For example, a management and development software for RDFS, OWL 2 (RL), F-Logic and Objectlogic is OntoStudio[9], supported by the semantic middle ware and inference machine OntoBroker[10].
In the context of ONTORULE (*cf.* section 2.5), a BRMS is able to draw on a KB in the form of an ontology that stores the facts (ABox) and conceptual data (TBox). Rule engines take a (derivation) rule set and the facts as input and interpret them in order to perform Create, Read, Update, Delete (CRUD) operations on the ABox, i.e., they create a set of assertions (Wood 2010). Usually, they do not alter the rules themselves or the ontology's TBox.

[6] http://www.drools.org/ (visited on 08/05/2015)
[7] http://www.jessrules.com/ (visited on 08/05/2015)
[8] http://www-01.ibm.com/software/integration/business-rule-management/jrules-family/ (visited on 08/05/2015)
[9] http://www.semafora-systems.com/en/products/ontostudio/ (visited on 08/05/2015)
[10] http://www.semafora-systems.com/en/products/ontobroker/ (visited on 08/05/2015)

Besides, management software for rules and ontologies usually feature various other vital functions, such as model testing, conflict resolution, constraints and verification, i.e., it checks "whether a proposed domain-specific model is in accordance with all the rules of the generic conceptual model" (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). The validation of the model, on the other hand, "is the process to check with the stakeholders (in their preferred language) who have the knowledge about the model, either implicit or explicit, that the proposed model is in accordance with the business reality" (ibid.).

2.5. ONTORULE Methodology

The ONTORULE Methodology (Berrueta et al. 2011; de Sainte Marie, Escudero, and Rosina 2011), a coinage meaning "Ontologies meet Business Rules", is an approach for the development of rule-based applications, combined with ontologies. It enables the various user roles to interact with the part of the business application that is most relevant to them by separating domain knowledge, business logic and application code. Throughout the development of our major case studies, we applied parts and ideas of this methodology. Furthermore, we used this approach, among others, for creating the architecture, ontologies and rules in the main part of this dissertation in a refined and customized way (*cf.* chapter 5). In this section, we only give a very rough overview on the ONTORULE approach, because the more detailed procedures are commonly described at the appropriate sections.

The main activities of the methodology are depicted in Figure 5.2: beginning with the *KA and modeling* activity, which transforms business and domain knowledge and logic, respectively, into formalized representation formats, through *management and authoring* of Semantic Web Technologies (SWTs), i.e., ontologies and rules, to the *execution* of this business and domain knowledge and business logic in an application.

Various user roles, i.e., senior business experts, business analysts, DEs, IT experts and operational users, cooperate and apply the methodology in the mentioned phases. They have been named in order to represent personae (*cf.* section 6.3.2 and Bonis and Bellino (2011)) and supported the development of our application by providing assistance for the application's usability (ibid.).

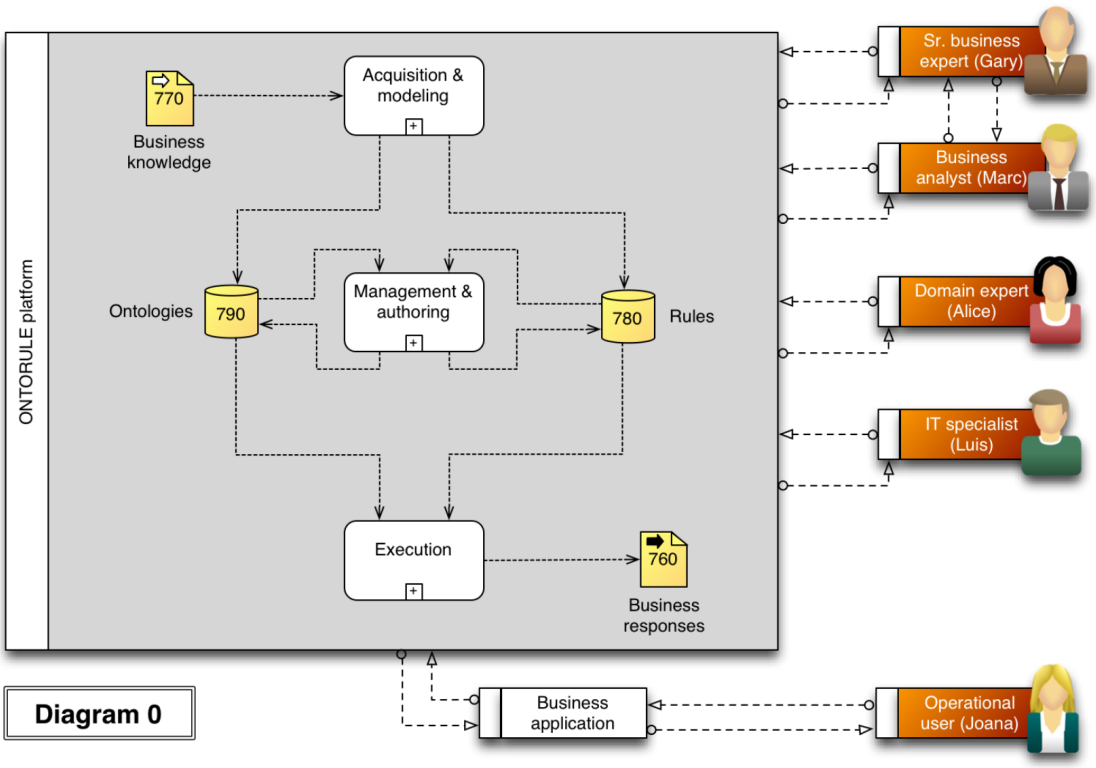


Figure 2.7.: The ONTORULE platform: illustrating three main activities and separation between ontologies and rules (Berrueta et al. 2011).

2.6. Virtual Product Development

Our business case throughout this thesis originates from the product development domain, i.e., our ontologies, rules, queries and the subjacent methodology are designed and modeled in order to support this area. More precise, our running examples and case studies often refer and relate to an automotive Research & Development (R&D) domain.

Therefore, this section provides basic knowledge about product lifecycles in general, classifies the (virtual) product development and introduces several primary technical terms.

Figure 2.8 helps to classify various relevant design stages and development processes in a product lifecycle. The business domain we are interested in is outlined by a red ellipse, namely, the product development which is implemented by a *"Product Development Process"* (PDP). Common steps during a PDP are the idea generation, screening, business analysis, concept and embodiment design, prototype (virtual and physical) and market testing, detail design and finally the

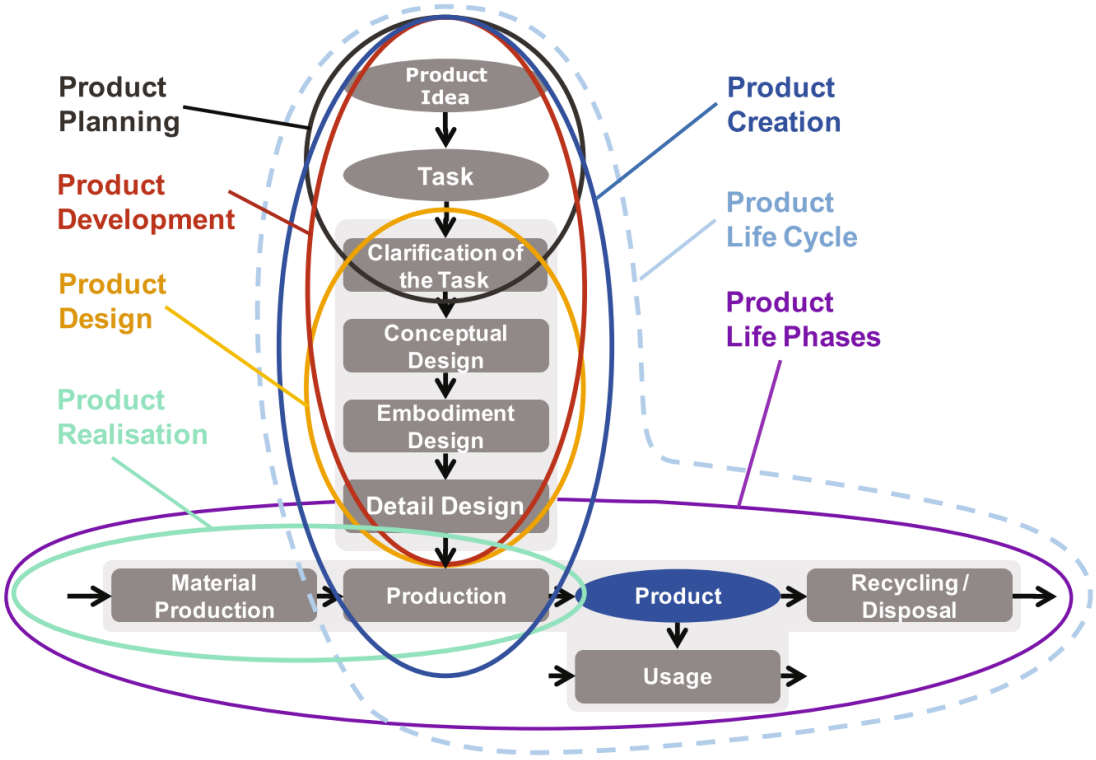


Figure 2.8.: The different phases of a product, i.e., the product lifecycle and design work (Birkhofer 2011).

product launch preparations. These process phases, especially their stakeholders, importance, manifestations, concurrency, dependencies and order, are individual to a company and several different state-of-the-art PDPs exist. Naturally, a plethora of enterprise divisions, for instance, marketing, R&D and finance, have to cooperate and collaborate in order to create a new product which makes this process even more complex. A universal important process milestone is the *Start of Production (SOP)* which is the release of a product into production, i.e., the handover between the R&D and production divisions. However, the PDP continues, because stakeholders do post launch evaluations and begin to plan and realize improvements for future product version, i.e., facelifts in the automotive domain, which start a new, pruned and customized PDP.

Nowadays, a lot of IT systems support the PDP and *Product Lifecycle Management (PLM)* in general. When these systems support the process extensively, we use the term ***virtual*** *product development*, *creation* or *lifecycle* (*cf.* Figure 2.8) that make use of the discipline of *Virtual Engineering*.

Definition 2.3 (*Virtual Product Creation*)

Continuous IT support during product and production creation with an intense application of simulation, validation and verification techniques on the basis of realistic models. The goal is to attain early product and production knowledge, a premature detection of product properties, as well as a reduction of physical prototypes (Eigner and Stelzer 2009b, trans.).

Definition 2.4 (*Virtual Engineering*)

Virtual Engineering *is the* timely, continuous, interconnected *(process view) and* integrated *(system view) support of the Product Creation Process (PCP) concerning* coordination, assessment *and* ascertainment *of the development results of all partners with the aid of virtual prototypes (Ovtcharova 2009, trans.).*

Example software solutions supporting or carrying a virtual PDP or even a virtual *PCP*, highlighted by the blue ellipse in Figure 2.8, are authoring tools, solutions for simulations and calculations, software for planning and simulating production and assembling as well as cooperative applications (Eigner and Stelzer 2009b). As the case may be, the *Computer Aided Design (CAD)* and *Computer Aided Engineering (CAE)* tools we describe in this thesis belong to this class. The physical mock-ups and prototypes that cannot be reduced due to regulations or functional obstacles, are heavily supported by IT systems or even a hybrid, like Hardware in the Loop (HiL), and relate directly to the introduced *Computer Aided Testing (CAT)*.

In order to manage these systems, solutions and the necessary data, most important product data, enterprises utilize *Product Data Management (PDM)* solutions during the PDP and PLM applications during the entire lifecycle, occasionally already supported by ontologies (*cf.* Matsokis 2010). Because product data of vehicles, vehicle parts and the associated product structure is used during the whole lifecycle by many organization units, they have to be pervasive and standardized. Therefore, they are usually based on the *STandard for the Exchange of Product model data (STEP)* (ISO 2014, AP 214) in the automotive domain, which is also applied and explained in our method ontology in section 4.6.1.

Developing products virtually bears many advantages in comparison to the conventional physical development as depicted in Figure 2.9.

Enterprises that apply virtual development and production commonly benefit in a number of ways, but mainly, the applications of the very cost and time intense

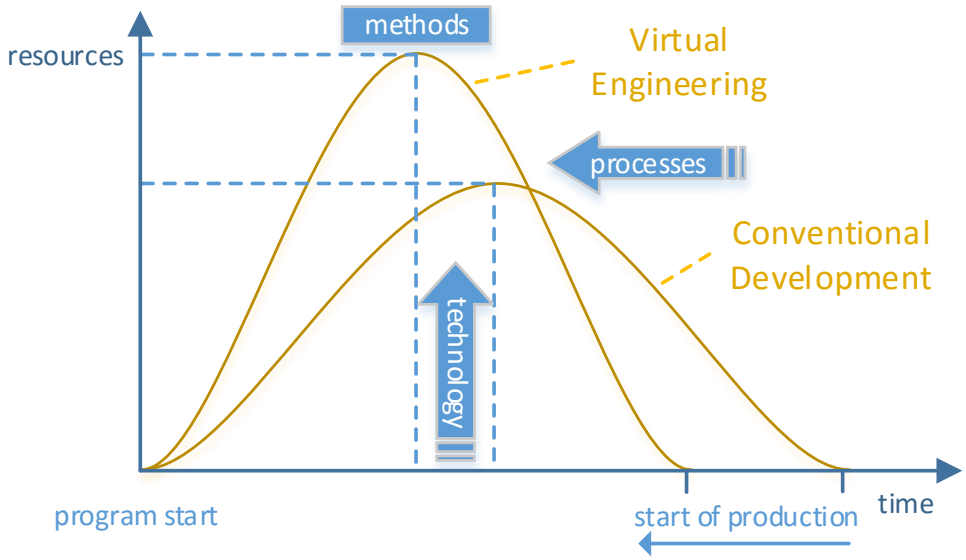


Figure 2.9.: Shifting of result critical sub-processes and resources towards early phases of development. According to Eigner and Stelzer (2009b).

physical simulations with real prototypes can be reduced by substituting them with Digital Prototypes (DPs) which in turn leads to a considerably shortened PDP, as illustrated and hence, we have an earlier SOP and market launch (Eigner and Stelzer 2009b).

However, this shifting is only feasible with an increased application of new technology, (virtual) methods and resources. As a consequence, our business motivation for this dissertation.

Furthermore, virtual prototypes, often called *DPs*, are often cheaper than physical ones, especially when performing lots of crash simulations. Besides, the simulations can be set up, modified, repeated and optimized more simply and their execution time is usually decreased. Moreover, a virtual model can be shared, reused and compared among divisions, suppliers and partners globally and instantaneously.

DPs represent a virtual, often executable, model of a usually former physical prototype. While a model is by definition a pragmatic and reduced mapping to a real system, nowadays, some physical systems are not constructed at all anymore, i.e., they only exist virtually (they are still models, though). The model's purpose is usually a validation or verification of an assembly's form, function and behavior

that can be achieved by running sundry simulations, e.g., Finite Element Method (FEM), Multi Body Simulation (MBS) or Computational Fluid Dynamics (CFD).
One important term associated with DPs is *Digital MockUp (DMU)*, significant in section 7.3, which represents a 3D model of an optimally complete product – DPs embrace many more facets, i.e., they correspond to a wide range of assemblies, areas and disciplines of the development, for instance, they involve kinematics, product structure, electr(on)ics, lighting, sound, driving behavior, mechanical strength and many more aspects.
Next to these disciplines, the virtual product development includes several other ones, for instance, the application of Virtual Reality (VR) or Augmented Reality (AR).
For further information on DPs and (virtual) product development in general, please resort to this section's various sources (Ehrlenspiel and Meerkamm 2013; Eigner and Stelzer 2009c; Ovtcharova 2009; Iparraguirre 2013; Autodesk 2007).

2.7. Enterprise Architecture Management

Many enterprises' IT landscapes have reached a degree of complexity that is only hard to understand and manage (Hanschke 2013). Additionally, the IT needs to be aligned to business services and processes for an optimal and efficient support. For this purpose, many organizations have established an "*Enterprise Architecture Management*" (EAM), motivated by the circumstance, that business is changing faster and faster due to shorter development cycles, adaptation and reorientation of business models and an overall need for improving the business and IT alignment.
In *The Open Group Architecture Framework (TOGAF)* standard, a popular *Enterprise Architecture Framework (EAF)*, described in the following section, an *enterprise* is defined as "any collection of organizations that has a common set of goals" (The Open Group 2011). That means, an enterprise can by anything, from a governmental organization, to a corporate group or a division inside a company or even just single department. So, an *Enterprise Architecture (EA)* can either cover a whole group or just a small fraction – the overlap is the coverage of "multiple systems, and multiple functional groups within the enterprise" (ibid.). The term *architecture* is also defined by the *Open Group*, based on *ISO/IEC 42010: 2007*.

Definition 2.5 (*Architecture*)

1. *"A formal description of a system, or a detailed plan of the system at component level to guide its implementation" (The Open Group 2011).*
2. *"The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time" (ibid.).*

EAM helps companies to reduce costs for maintenance and developments when confronted with an increasing number of systems by analyzing "areas of common activity within or between organizations, where information and other resources are exchanged to guide future states from an integrated viewpoint of strategy, business and technology" (EABOK Consortium 2014).
Success factors are a goal-oriented adaption towards changing market and boundary conditions, the detection of redundancies (Auer et al. 2011) and the definition, determination and an ongoing assessment of the various EAM Key Performance Indicators (KPIs) (ibid.).

Definition 2.6 (*EAM*)

"EAM is a systematic and holistic approach in order to understand, communicate, design and plan the professional and technical structures in enterprises. It supports further developing the IT landscape strategically and in a business-oriented way by making the IT landscape's complexity manageable. [EAM] is an essential component of the strategic IT management and includes processes for the documentation, analysis, quality assurance, planning and controlling of the IT landscape's and business architecture's further developments." (Hanschke 2013, p.3, trans.)

"Above all, [EA] recognizes that the information assets of an enterprise are always in flux, and that this flux is the steady state" (Wood 2010).
The business highly depends on a working and manageable IT architecture as a basis for efficient business processes hence these two domains have to be modeled and aligned when following this approach.
An EA's centerpiece is its meta model which describes an enterprise's central artifacts, as well as the artifacts' relations (Aier, Riege, and Winter 2008). It represents the diverse required aspects of a company's IT and business structures (*cf.* Figure 2.10). These are, for instance, elements such as processes, services, organizational structures, data, applications and technologies. They are connected and orga-

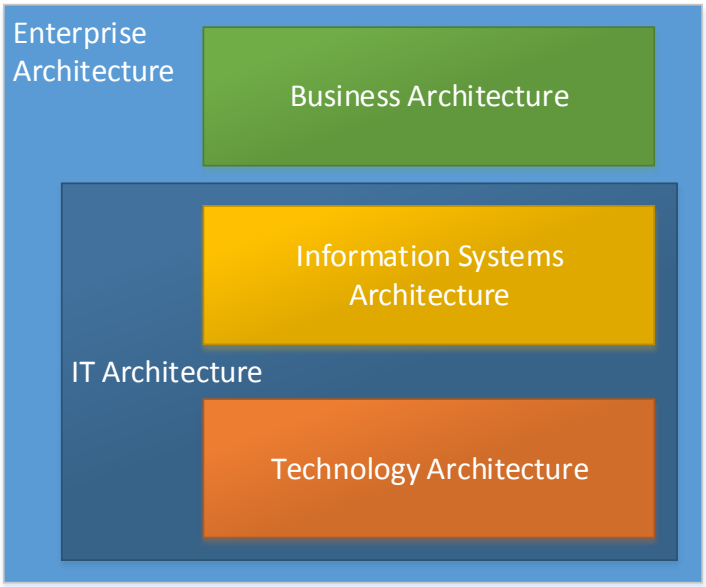


Figure 2.10.: Enterprise Architecture domains. Based on Weinberger (2010).

nized via sundry relations and clusters which form the EA. The IT Architecture can be further subdivided into architectures covering software and technology.
Every company is unique, with diverging requirements, goals and focus areas. For this reason, a suitable, customized meta model is more important when introducing an EAM in a company than a tool which comes along with an inflexible standard meta model.
This meta model along with suitable tools and an EAM methodology allows the enterprise to document and monitor the business and IT architecture in a holistic way which enables the architects and managers to react faster to occurring changes in an agile enterprise. The architectures and the containing data can be analyzed in order to comprehend and communicate coherences. A typical analysis answers queries about, for instance, the relationship between business services, which interfaces are used, which data is required, who uses which process or service and which business service supports the overall business strategy. The results can be processed to fit a target audience by providing views geared to the target group, such as managers, developers or DEs. The architecture describes all the enterprise information assets and helps to find the business processes in which these assets are used. Furthermore, it supports the identification of ownership of data and thus makes this data better maintainable and manageable.

EAM lets the company control the development in order to design the IT landscape strategically, e.g., migrate or synergize existing IT systems. Typically, this is done by planning from a *baseline architecture* to a *target architecture* (*cf.* Gleichauf 2011). For example, a portfolio management allows launching projects with minimally overlapping requirements and distinct rules.
Furthermore, modeling the business and IT architecture leads to an increased transparency which in return drives the use of a common vocabulary in the company and hence reduces misunderstandings. Other benefits are the advancement of standardization, an improvement and assurance of quality, a reduction of IT costs and consequently an improved coping with risks.
Generally speaking, the above listed advantages should provide a fillip for any organization. However, introducing an EAM is especially worthwhile for big enterprises. Such an enterprise can even be an "extended enterprise", that "nowadays frequently includes partners, suppliers and customers", as well as internal business units (The Open Group 2011). On the one hand, small organizations shun the undertaking, because it is wedded to a lot of effort – time- and resource-wise. On the other hand, a big enterprise, with a historically evolved IT landscape and complex business processes, will see the most benefits from such an endeavor.
In this thesis, we mainly focus on the business level and its relations to the information systems, for instance by dealing with business strategy, processes, organizational structure and business capabilities and the applications relevant for the execution of the business processes.

2.7.1. Enterprise Architecture Frameworks

An EAF describes a methodology for developing an EA and its use during operation. It usually includes definitions, models, best practices and describes the involved roles and process phases that act in the different EA domains, *viz.*, the business, data, application and technology domains.
Historically, EAFs originated with the "Zachman EA Framework" in 1987 (Zachman 1987), a compendium of guidelines for building an IT architecture.
It points out the relevant aspects and focuses that an enterprise should consider when creating information systems. The focuses in the guideline are subdivided into the fields data, function, network, people, time, and motivation. In combi-

nation with the six player perspectives planner, owner, designer, builder, subcontractor, and enterprise, the EAF describes best practices for many situations when creating an IT architecture. This way, each viewpoint of a system is considered from the perspective of every stakeholder in the enterprise. However, the Zachman Framework, in contrast to the majority of modern EAFs, does not include a methodology for creating this architecture.
Some examples for EAFs are *TOGAF* (The Open Group 2011), which is described in the next section, *ISO 19439:2006* or the *US Federal Enterprise Architecture Framework*, to name just a few (Matthes 2011).

2.7.2. The Open Group Architecture Framework

In 2011, the Open Group released the most current TOGAF version 9.1 (The Open Group 2011) – its initial development began in 1995.
The framework provides a methodology for the design, planning, development, implementation and maintenance of an EA. It does not feature a specific model the companies can use, but offers meta models and guidelines that should be applied as seen fit and as required.

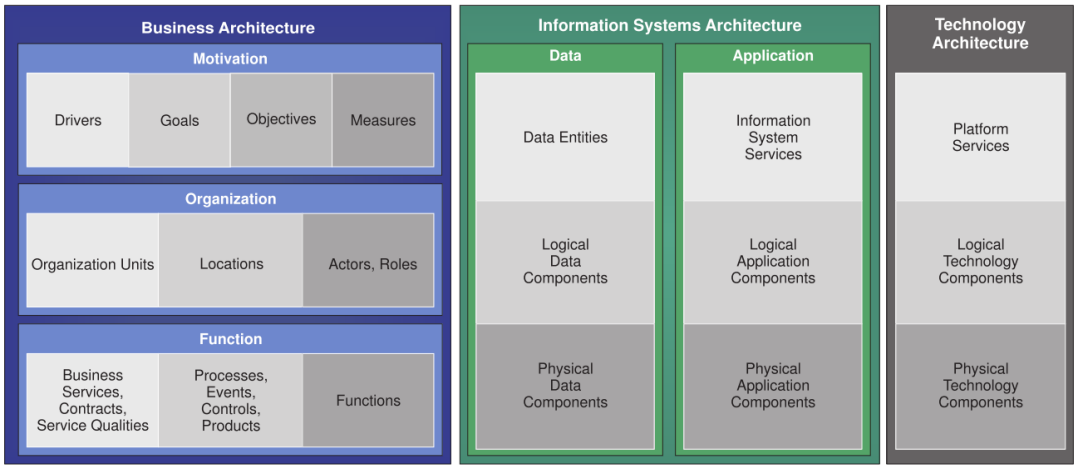


Figure 2.11.: TOGAF Content Metamodel (The Open Group 2011).

TOGAF partitions an EA into four main architecture tiers (*cf.* Figure 2.11).:

Business architecture: The first architecture tier deals with business strategy, business processes, governance, organizational structure and business capabilities.

Data architecture: The *data architecture*, i.e., the data model which is part of the *Information Systems Architecture*, holds the information about physical and logical data required and produced by the business processes, including their relationships and definitions.

Application architecture: The applications relevant for the execution of the business processes are modeled in the *application architecture*. It consists of interface descriptions, deployment status and relationships between those applications and a business level description of their functionality. It is classified into the *Information Systems Architecture*, as well.

Technology architecture: The bottom tier architecture holds information about the IT infrastructure, like servers, middleware and networks, which is the requirement for operating the upper tiers.

The TOGAF Content Metamodel in Figure 2.11 illustrates these four main architecture tiers, vertically divided in the middle section. It presents the "building blocks that may exist within an architecture" (The Open Group 2011) and how they can be related to each other.

The adaption of the TOGAF to an organization's needs is described in the TOGAF Architecture Development Method (ADM): A methodology for developing and governing an EA that fulfills the enterprise's requirements based on TOGAF building blocks. It is an iterative approach, partitioned into different phases (*cf.* Figure 2.12).

The cycle begins with the preliminary phase, where the introduction of a new or changed EA is prepared and initialized by adapting the TOGAF model to the enterprise's needs and by choosing *architecture principles*. In the following Phase A (Architecture Vision), *architecture goals* are defined and stakeholders identified. The corresponding business architecture is then planned in Phase B. The aforementioned *Information Systems Architecture* is prepared in Phase C, followed by an appropriate Technology Architecture in Phase D. Guidelines for the implementation of previous phases are defined in Phase E (Opportunities and Solutions). The subsequent phases F–H deal with the planning of migrating from a baseline to target architecture (Migration Planning), the implementation of governance and the definition of processes for this migration in the Architecture Change Management.

The ADM's centerpiece is the Requirements Management, that is connected to all the phases.

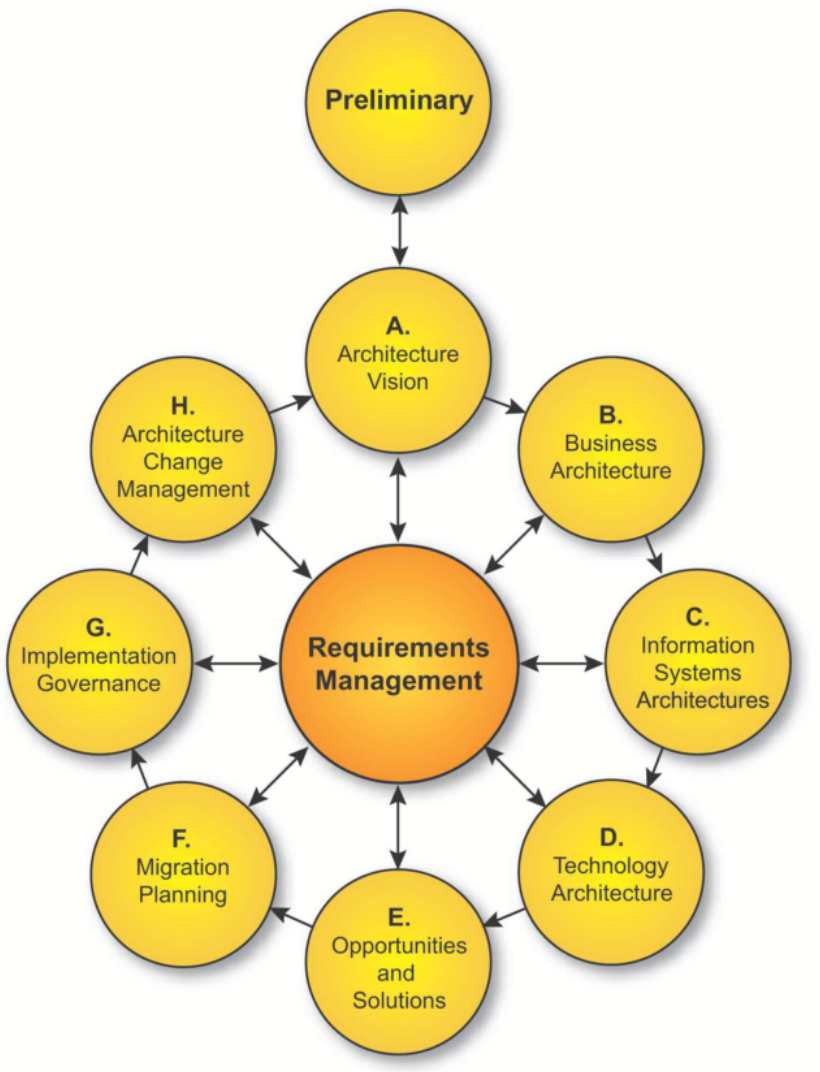


Figure 2.12.: ADM Cycle (The Open Group 2011).

2.7.3. ArchiMate

ArchiMate (The Open Group 2013) primarily provides a uniform graphical representation language for EAs. The standard offers different kinds of meta models, diagrams and viewpoints in order to model a tailored graphical representation for the involved stakeholders. However, we do not use this graphical notation in this thesis, because we mainly model ontologies and not solely EAs. Nevertheless, we reuse meta models and aspects provided in the standard.
The ArchiMate standard draws heavily on TOGAF vocabulary as it is also maintained and developed by The Open Group. The mapping, depicted in Figure 2.13, exemplifies this interleaving.

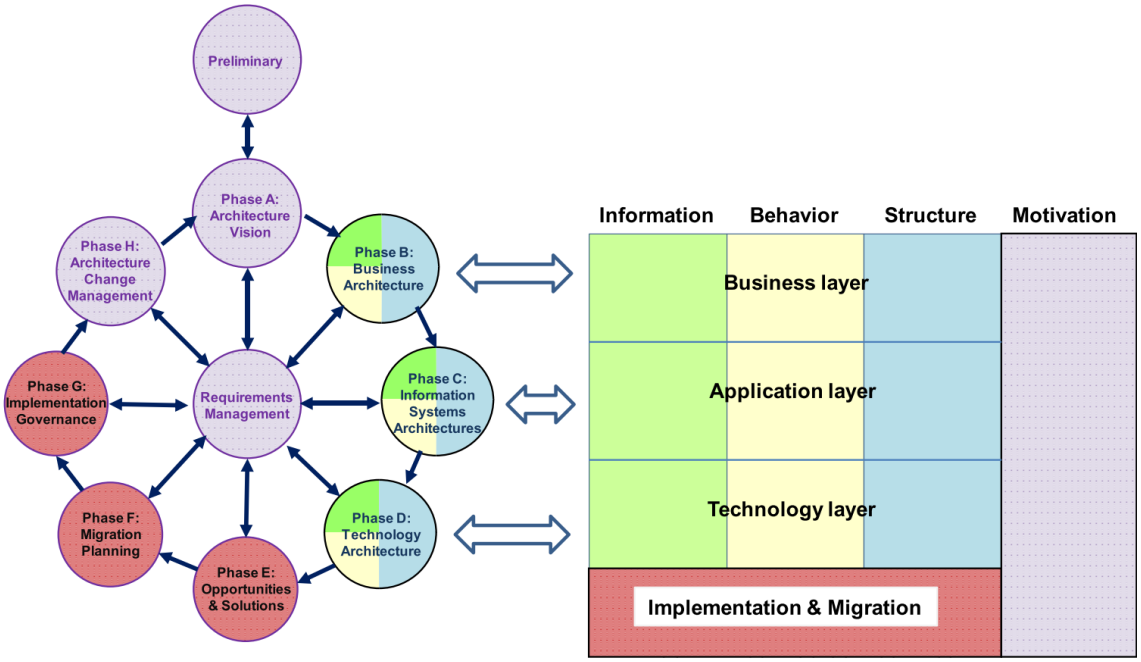


Figure 2.13.: Correspondence between ArchiMate (including extensions) and TOGAF (The Open Group 2013).

In the figure, ADM phases are linked to the ArchiMate Framework. Horizontally, the framework is divided into the well-known layers: Business, Application and Technology. An addition is the ArchiMate Motivation Extension, whose elements "provide the context or reason lying behind the architecture of an enterprise" (The Open Group 2013). Vertically, it distinguishes between passive structure aspects, i.e., information concepts, behavior and active structure aspects, like processes. Furthermore, ArchiMate offers the Implementation and Migration Extensions that covers the ADM phases E–G of the methodology.
Besides the aspects shown in the previous figure, The Open Group (ibid.) acknowledges that an EA concerns many more business domains, such as:

- Goals, principles, and requirements
- Risk and security
- Governance
- Policies and BRs
- Costs
- Performance
- Timing
- Planning and evolution

It does not provide views and models for every listed domain above, but allows extending the ArchiMate language in order to cope with them. Sometimes, there is no one-to-one mapping between TOGAF and ArchiMate, but they "can easily be used in conjunction and they appear to cover much of the same ground, although with some differences in scope and approach" (ibid.).
TOGAF, together with ArchiMate, serves as the foundation for our EA meta model, which is introduced in chapter 6.

"Out of clutter, find simplicity."

Albert Einstein (1879 – 1955)

3

Related Work

3.1. Synopsis

In this chapter, we introduce or point to related work concerning the domains dealt with in this thesis, i.e., selected papers concerning the design of method frameworks, metrics and modeling standards that match chapter 4; the integration of the method ontology into an Enterprise Architecture (EA), the different views and roles and the selection, adaption, analysis and application of methods and their related frameworks applying to chapter 6. Furthermore, we underline the differences to the presented approach in this thesis.
Besides, we will present method descriptions and definitions, although, most of the relevant related work concerning this topic is included in the first sections of chapter 4. Further information and comparisons about different method definitions and models in academia and industry can be found in C. Braun, Hafner, and Wortmann (2004) or Weigt (2008), though.
As mentioned, our methodology is based on the ONTORULE approach that has already been briefly introduced in section 2.5, which includes references to related work and state-of-the-art. Besides, additional relevant and similar sources, like methodologies about ontology development, Knowledge Management (KM) in general, applied standards and software development for and with Semantic

Web Technology (SWT), have been pointed out in chapter 5 and thus will not be revisited here.
Finally, we will conclude this chapter by comparing the presented papers among themselves, but mainly with this thesis' solution and hence we present its unique characteristics.

3.2. Method Frameworks and Meta Models

Process-oriented Method Model Birkhofer et al. (2002) have created the method framework *Process-oriented Method Model (PoMM)*, depicted in Figure 3.1, in order to optimize a product development method's description. Their work has been motivated due to the fact that method frameworks for the design in a Product Development Process (PDP) had not been available, resp., the existing ones had mainly a very specific focus and were created for academia mostly. Therefore, they collaborated with other technical universities in order to generate a method pool that is standardized, extensive and thorough. For example, the educed meta model has been applied in (Ernzer and Birkhofer 2002), as described in section 6.4.
They bisected their model whereas the top part in Figure 3.1 is about accessing or selecting the methods and the bottom area is describing the methods' meta model.
They regarded a wide range of involved stakeholder roles, fields of application, the methods' chronological order, their contextual information, possible working aids, hints and other influencing parameters, for instance, the infrastructure or user skills, for the methods' meta model. They even considered interrelationships between methods. Furthermore, PoMM allows describing a method's in- and output which allows linking methods to support entire processes. The method itself is described in a superficial way in the sequence block (*cf.* Figure 3.1) which can include a structural and textual procedure and description, resembling our `Procedure` concept, combined with the method representation options introduced in section 4.3.2.
The PoMM covers a wide range of our method ontology; as a matter of fact, some modules are exclusively available in the PoMM whereas others, like our concepts `Goal`, `Function` (*cf.* section 4.3) and our sophisticated relations to business processes by `Concrete Methods` (*cf.* section 4.4.2) are not regarded in the here in-

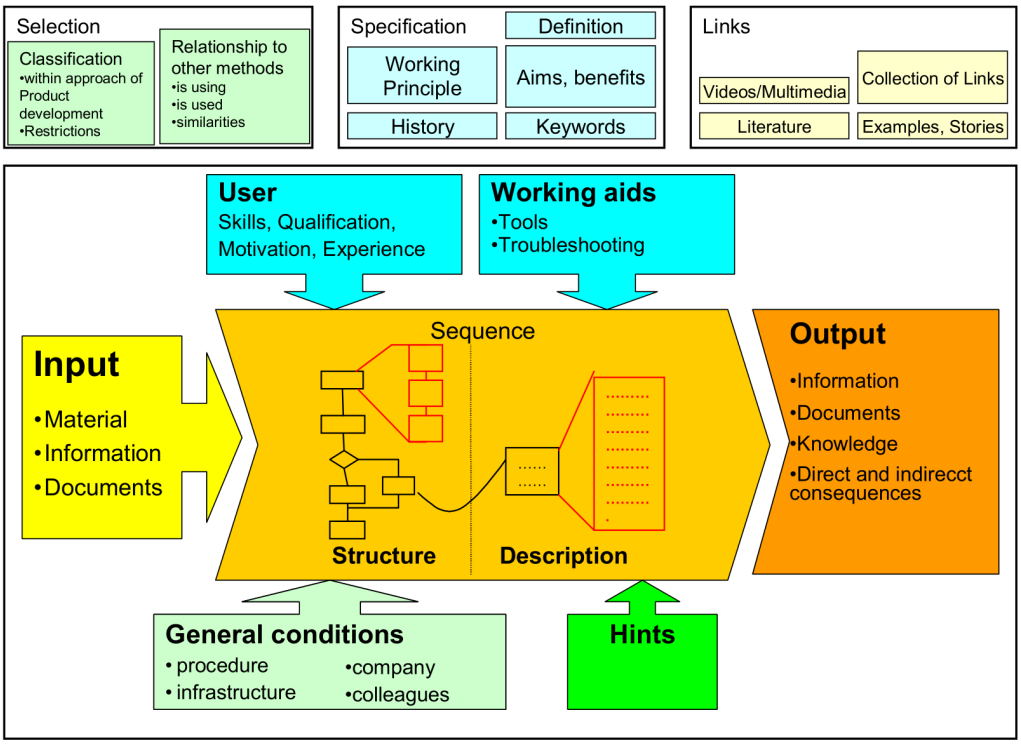


Figure 3.1.: PoMM (Birkhofer et al. 2002).

troduced meta model. In contrast to our formal meta model representation, they presented a conceptual model and ideas in their paper. However, as the meta model elements resemble our ontological concepts and due to the fact, that our ontology is easily customizable, an implemented PoMM representation can be mapped to our ontologies. Besides, our method ontology performs the part of an upper ontology (*cf.* section 4.6) hence, should be customized and supplemented with Domain ontologies (DOs), anyways.

Munich Model of Methods (MMM) T. Braun and Lindemann (2003) have analyzed the selection, adaption and application of product development methods for the impersonal transfer of method know-how. The results act as the foundation for a method model which consists of method building blocks (*cf.* Figure 3.2), linked by method attributes. Furthermore, the model supports implementation of *superior tasks* and *resources* and *support* (*cf.* Figure 3.3) can be related to a procedure. Their outcomes concerning the method selection phase are introduced in section 6.4.2 – here, we focus on the underlying model in order to compare it with our solution.

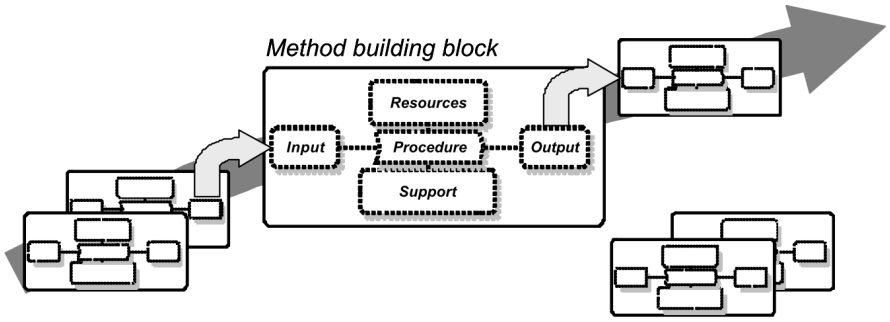


Figure 3.2.: Method building blocks (T. Braun and Lindemann 2003, extract).

Method building blocks can be combined to a network, like method alliances and method chains introduced in section 4.2.1, because a methods' output often directly refers to the next method's input.

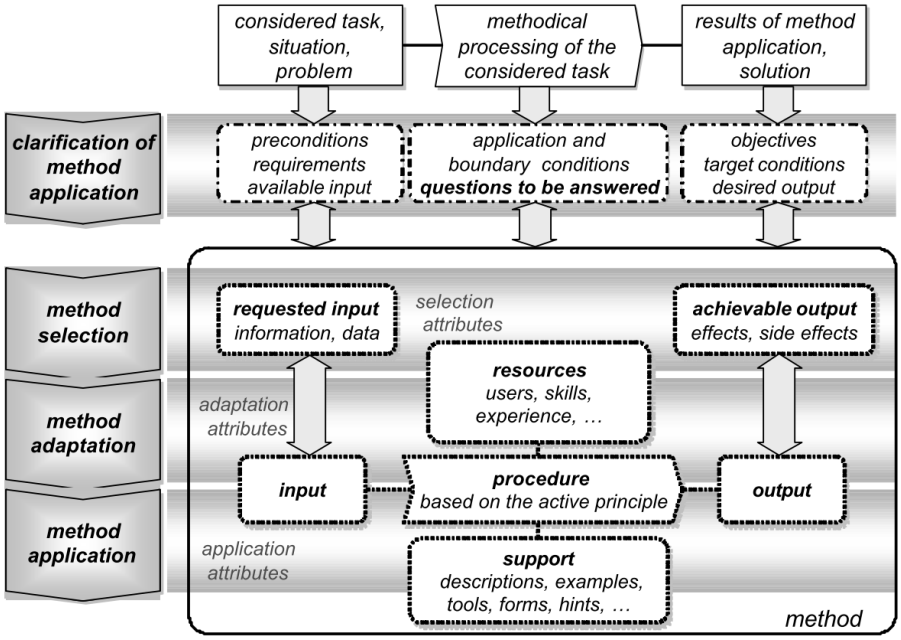


Figure 3.3.: The *MMM* (T. Braun and Lindemann 2003; Lindemann 2009).

The diagram by T. Braun and Lindemann, depicted in Figure 3.3, is horizontally split into four phases. It juxtapositions method attributes and the method implementation during the process action. The first lane deals with the clarification of the method's use case scenario. The following lanes represent the method selection, its customization and its application.
Vertically, the model is divided into different process building blocks, i.e., the requirements phase, boundary conditions and questions concerning the method

application and of course the method's goal and output in the third vertical lane (T. Braun 2005).
The single phases are supported by "tools", like methods descriptions, examples, forms or implements/tools.
The majority of the depicted blocks are also covered by our method ontology, because the entities concerning a method definition (see section 4.2), like input, output, procedure etc., necessarily have to be implemented in a model when dealing with working methods. On the other hand, the large overlap is hardly surprising, because the MMM has also been an inspiration when we created our own model. In opposition to the MMM, we only marginally dealt with method customization. Besides, we will react to the block considering user experiences, capabilities and roles as a resource, when choosing a method for a task. We will deal with this knowledge in chapter 6 about Enterprise Architecture Management (EAM) integration.
Independently from the methodology, which is the focus of the MMM, we extended our model with ontologies covering metrics, business processes, a Controlled Vocabulary (CV) etc. Furthermore, we present a possible implementation with SWTs which is a key for answering the queries concerning the method clarification, selection and application for a wide group of users.

Competence in Design and Development (CiDaD) Portal The Institute of Product Development of the Technische Universität München (TUM) developed the CiDaD Portal[1] on the basis of the Munich Approach Model (*cf.* Lindemann 2009). It is a web platform (*cf.* Figure 3.4) that supports users that are designing, developing or learning about technical products and the fundamental design processes.
The platform offers a collection of various method descriptions, tools, aids and modular and digitalized books. Furthermore, the user has access to attachments in terms of forms and presentations for the domain of product development. The user can navigate through the different articles and a glossary via the hyperlinked texts in the book articles, lists or by using the search functionality. Besides, a selection can be made by picking superior processes. All the contents originate from the institute's projects as well as its published books.

[1] http://www.cidad.de/; last visited on 12/17/2013

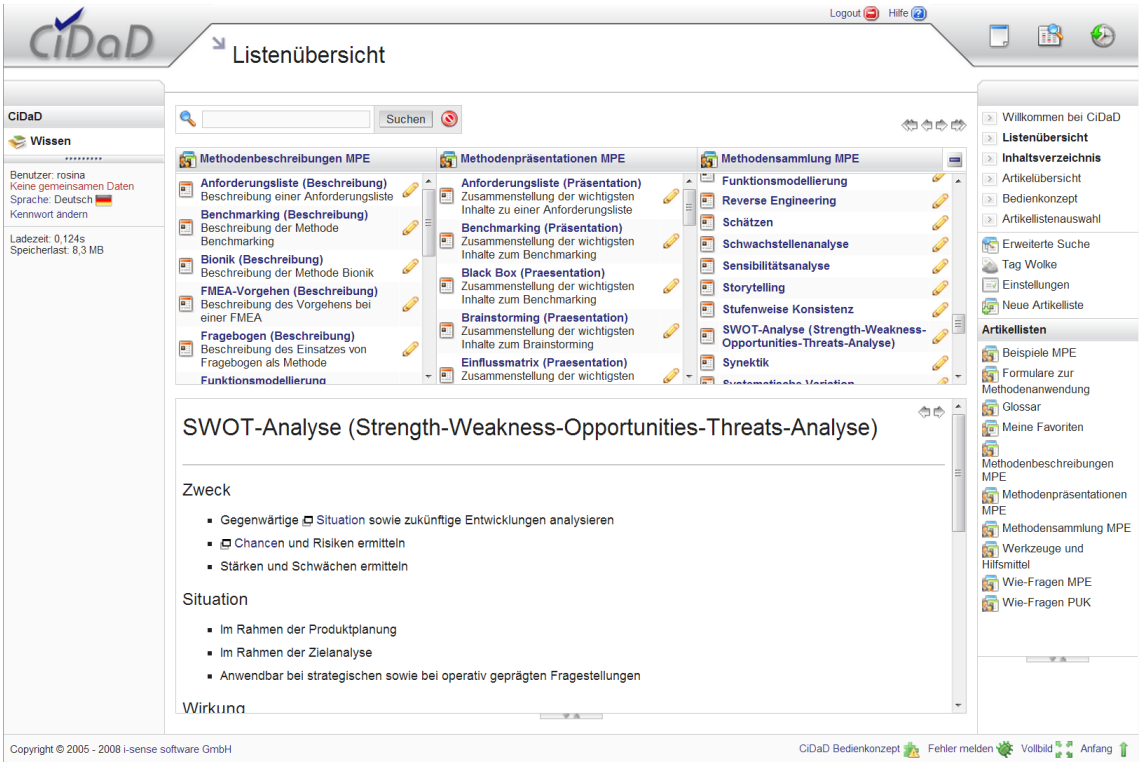


Figure 3.4.: Exemplary screenshot of the CiDaD portal.

As we can see, the CiDaD Portal is a web-based method database, offering mostly written articles about product development. Methods can either be selected by narrowing the search window with the support of the superior processes or by entering keywords in the search field. While this technique allows finding methods by their contexts, e.g., based on their input or output information, it does not classify the various terms, nor does it support searching for synonyms or super-/sub-concepts. Besides, fundamental method attributes, e.g., purpose, situation, effect, procedure, tools and hints, are just written down textually and not part of a meta model. Their approach allows a straightforward extension of their content by adding new web pages and texts, whereas our approach is more sophisticated. Thus our approach is more time-consuming when adding new contents but therefore allows a more fine-granular selection. Besides, the ontologies cover a broader range of concepts.

Methodos The method model kit *Methodos* (Franke, S. Löffler, and Deimel 2003) is an internet portal and assisting system of methods, i.e., a collection of sundry design methods for the whole PDP. The data of the method model kit is stored in

a database which is accessible through a web front end. They are ordered by their field of activity, e.g., product planning, gathering of ideas or concept assessment (Stefan Löffler and Jagusch 2004). A finer grained distinction is offered by different method classes which are structured hierarchically. Beyond that, the method model kit offers detailed and structured descriptions about the method itself, its advantages and disadvantages, necessary qualifications for the user, practical examples and additional literature (Weigt 2008).

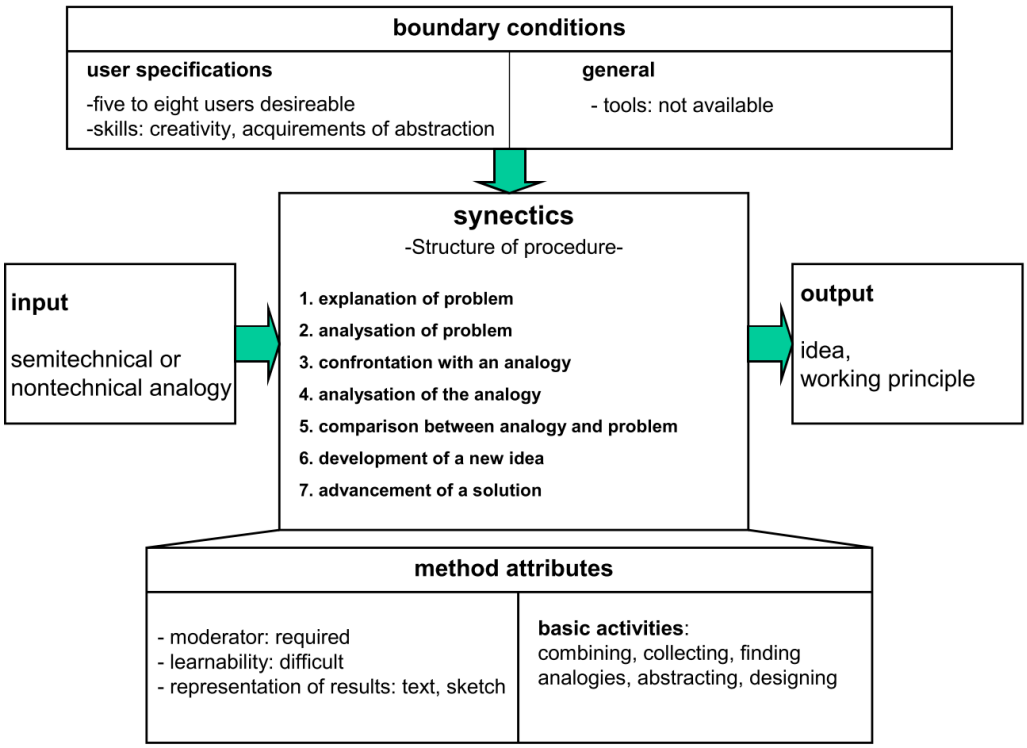


Figure 3.5.: *Methodos*: Method synectics and stored information (Franke, S. Löffler, and Deimel 2003).

The underlying meta model, depicted in Figure 3.5, illustrates Methodos' synectics, boundary conditions and a selection of method attributes. Like the method definition chosen in this thesis, the in- and outputs are specified as well, which allows the user to combine several methods. In contrast to our method framework, the methods' contexts are not described formally, i.e., inputs and outputs, tools and goals are described in textual form inside the method description. The modeled method attributes and relationships, like variants, preconditions, alternative methods, tools etc., are represented as Strings in the database, i.e., they are not interconnected and thus cannot be queried directly.

MAP-Tool The MAP[2]-Tool[3], developed by the University of Karlsruhe, assists the user in selecting appropriate tools and methods for a specific goal (T. Braun 2005). The methods are related to a superior product innovation process, segmented into phases, process steps and activities. It has been mainly designed for Small and medium enterprises (SMEs), illustrating a short method description and its requirements as a teaser in the search results window. A more detailed description represents information about the method's implementation, strengths, weaknesses as well as further literature. Furthermore, it provides a short slide show for each method in order to quickly generate a basic method understanding.
This Internet portal has many commonalities with the previously introduced Methodos and CiDaD Portal. It is a front end for users to search and select methods, suitable for a specific scenario. Likewise, the presented information roughly covers the known method contexts. In contrast to our approach, keeping in mind that it deals with a method model framework, which is based on semantic technologies, and not its User Interface (UI), the introduced MAP-Tool has a rigid model structure. Thus, it is not as easily extendable and does not cover the same set of concepts.

Method Selection and Combination in a Product Creation Process (PCP) Buchert et al. (2014) have developed an approach that proposes an appropriate method in a PCP to an end user as illustrated in Figure 3.6. Their focus and primary criterion for the method selection had been the product's sustainability: The end user can choose the desired method output by providing simple parameters. Subsequently, the expert system selects the best fitting methods from an underlying database.
In order to be able to choose a suitable method, they developed a method meta model, as well. Their meta model features the method attributes focus of method, addressed life cycle phases, point of application, dimension of sustainability, type of processed data, user of method and a layer of abstraction.
Likewise, they tested their approach in a Computer Aided *x* (CAx) environment, i.e., they used their system in order to provide method suggestions concerning the product creation of a turbocharger which was present as a Computer Aided Design (CAD) model as well as a physical entity.

[2] "vom Markt zum Produkt" – ("from market to product")
[3] http://imihome.imi.uni-karlsruhe.de/map.html; last visited on 12/17/2013

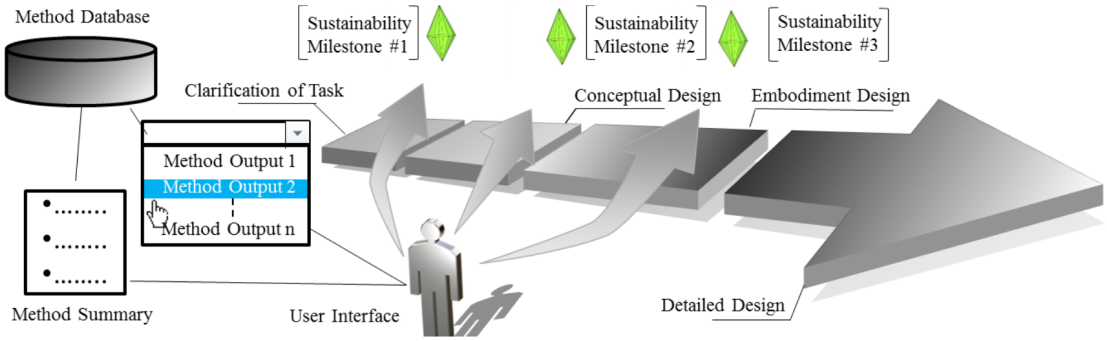


Figure 3.6.: Method Selection and Combination in a PCP (Buchert et al. 2014).

While in the majority of cases they use a different terminology, we can find many similar concepts compared to our framework, for instance, method attributes. However, they concentrate on sustainability in a PCP, use more conventional technology and miss a lot of features that we included, e.g., metrics and CV. Furthermore, the point of time for the method's possible execution is provided only in a rough way, i.e., for a process phase as depicted in Figure 3.6. Additionally, the method's in- and output are described in a way coarser way. This is done on purpose, though. Finally, the dedicated user roles in their framework involve only the method user.

3.3. Method Metrics

Quality Function Deployment Greiffenberg shows how to objectively assess methods using Quality Function Deployment (QFD) (Greiffenberg 2003). After defining the term method in the sense of an engineering method he introduces the QFD. This approach is based on a House of Quality (HoQ), a matrix representing customer requirements, associated with defined measurements by the product developer. He introduces various method quality attributes, e.g., the degree of formality, usability, analyzability, reliability, maturity and many more. These "ilities" are then weighted in the HoQ. Its aim is assessing, and hence developing and designing methods purposefully.
Greiffenberg's categories for the method assessment and the introduced formulas could create an added value for our method framework. In section 4.4, we did not examine the sundry quality attributes in detail, but suggested how maturity models and a metrics ontology's Terminological box (TBox) should look like

and how it should be related to the method framework. Applying the QFD and reusing the quality attributes introduced in the HoQ for these ontologies would create a sound and prudent basis.

Metrics Ontology in Business Process Management Pedrinaci and Domingue (2009) show, how SWTs "can increase to an important extent the level of automation" for the domain of Business Process Analysis (BPA). The introduced complex metrics ontologies are used in combination with a computation engine in order to support the whole life-cycle of Business Process Management (BPM). In their published work, the metrics ontology is very detailed, for instance, they show how to demarcate between different SI units, how to represent derived units, aggregations, functions, ratios etc. With the help of these units and scales they can define functional metrics for, e.g., "Return on investment (ROI)" or "Process Instance cost", thus "spanning from low-level monitoring details to high-level business aspects" (ibid.).
As stated above, when we created our metrics ontology, we wanted to offer an upper level framework that still has to be customized by the user, i.e., extended and specialized by relevant DOs. Likewise, in our ontology, the concepts and relations suitable for a specific company can then be used in order to compute high-level statements, as well. In contrast to the work of Pedrinaci and Domingue we suggest using a Business Rules Language (BRL) or query language, for instance, Rule Interchange Format (RIF) or SPARQL, for this task.

3.4. Modeling Standards

Software and Systems Process Engineering Meta-Model Software and Systems Process Engineering Meta-Model (SPEM) is a prominent Object Management Group (OMG) meta model for describing software development processes and software development methods (Object Management Group 2008). In contrast to our method definition, a method in SPEM represents a development methodology like the waterfall model or an arbitrary agile approach. Next to this method content, SPEM allows modeling processes (*cf.* Figure 3.7), independently from the methods which allows reusing both, the methods and the processes in other contexts.

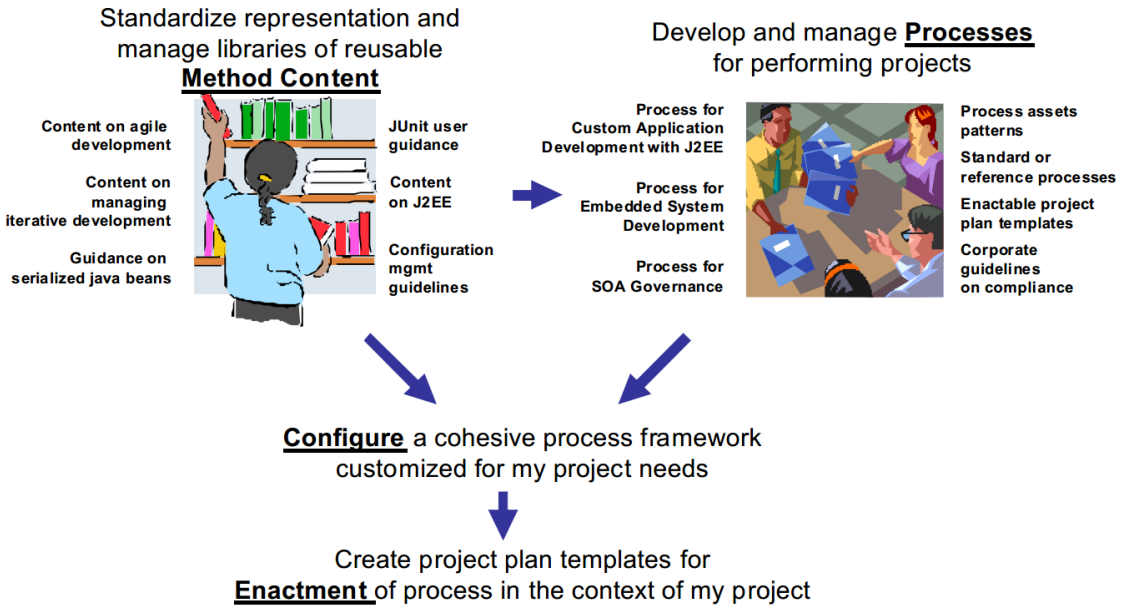


Figure 3.7.: SPEM's conceptual usage framework (Object Management Group 2008).

The meta model reuses Unified Modeling Language (UML) for the notation of its various elements, whereas we did not determine a fix notation for our method models. While we discussed the differences between processes, methods and methodologies at the beginning of this chapter, inarguable, many similarities exist.
Like our approach for the application of a working method, SPEM allows selecting and customize chunks and fragments of method(ologie)s and processes for a specific project.

ASAM ODS The industrial standard Association for Standardisation of Automation and Measuring Systems (ASAM) Open Data Services (ODS) (Bartz 2009) is used for managing measurement and test data, primarily in the automotive industry. This standard has been used as a basis for vendor-specific models, e.g., at Audi AG, that include method descriptions. Its main contributions are a common data model, Application Programming Interfaces (APIs), a physical storage standard, an exchange format and a set of typical usage scenarios. Whereas we concentrated on development methods in general, we find many similarities with the tests modeled in ASAM ODS. Therefore, a contrasting juxtaposition is presented in section 7.4 as part of our evaluation.

3.5. Integrating method knowledge in an Enterprise Architecture

Naturally, integrating new or modifying existing knowledge in an established EA is described in the various EA methodologies and Enterprise Architecture Frameworks (EAFs), e.g., TOGAF Architecture Development Method (ADM) for The Open Group Architecture Framework (TOGAF) (The Open Group 2011), the Systemic Enterprise Architecture Methodology (SEAM) (Wegmann 2003) for business and Information Technology (IT) alignment for SEAM-based EAs (Wegmann et al. 2007) or the best-practice-EAM by Hanschke (2013), to name just a few. An ancillary and good overview of the various EAFs in literature and current practices is given in Aier, Riege, and Winter (2008) and Matthes (2011).
The integration or mapping of Semantic Web (SW)-based Knowledge Bases (KBs) in general and our customized approach and related approaches will be dealt with in the following chapters exhaustively.

CAx Architecture The approach presented in C. Hess, Chen, and Syldatke (2008) is not directly concerned with an EAF, but deals with the integration of CAx methodologies into an IT and business architecture using ontologies. Their approach is revisited and refined in C. Hess, Chen, and Syldatke (2010). In C. Hess, Lambertz, and Syldatke (2009), the authors introduce another CAx architecture, depicted in Figure 3.8, that also features Computer Aided (CA) methods, as well as business objects, data, processes, functions, services, properties and applications.
For this purpose, they use SWTs, as well. In Figure 3.9, an extract of their ontology is illuminated which has been a spadework for our approach. The previous papers by C. Hess et al. introduce abstract ideas for the combination of the mentioned domains and an outlook which has been a motivating factor and groundwork for some concerns of this thesis, i.e., the idea of integrating CAx knowledge into an IT and business architecture. This is due to the fact, that we worked together in the same EA team and these papers reflect preliminary work to my approach.
However, my approach is based on an independent methodology (*cf.* chapter 5), the method meta model and its context presented in chapter 4 is not attached to their work and the EA integration introduced in chapter 6 is an independent de-

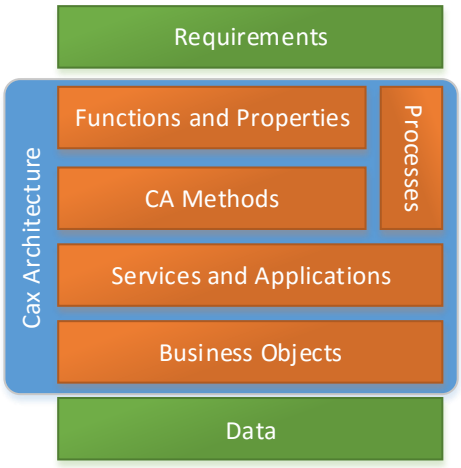


Figure 3.8.: CAx architecture layers. According to C. Hess, Lambertz, and Syldatke (2009)

velopment, as well. Taken all together, their work mostly influenced the thesis' motivation, requirements and the developed case studies.
C. Hess, Lautenbacher, and Fehlner (2013) introduce Business Building Blocks (BBBs), "an object that is in tight relationship with the business activities of an enterprise", for enterprise transformations, exemplified with a TOGAF-based EA. In contrast to domains, a BBB is even more fine-grained and hence can be discussed with more specialized stakeholders. We did not apply this concept, but mention it here, because it shows the more recent continuation of their work.

Multi-perspective Enterprise MOdeling (MEMO) MEMO, introduced in Frank (2002), provides techniques, heuristics, a process model and a set of modeling languages, i.e., Domain Specific Languages (DSLs), that are used to design enterprise models. This approach makes use of an ontology, as well. The research in this project is still ongoing (Frank 2014) and describes how methods can be integrated in an enterprise modeling approach. However, Frank focuses on modeling methods for *method engineering*, that consist of at least one modeling language and process model that guide users in the construction and analysis of models. They provide a method meta model, though, that features similar concepts to our method meta model as depicted in Figure 3.10. However, this meta model has a totally different structure, is realized with alternate approaches and technologies and is meant for another purpose.

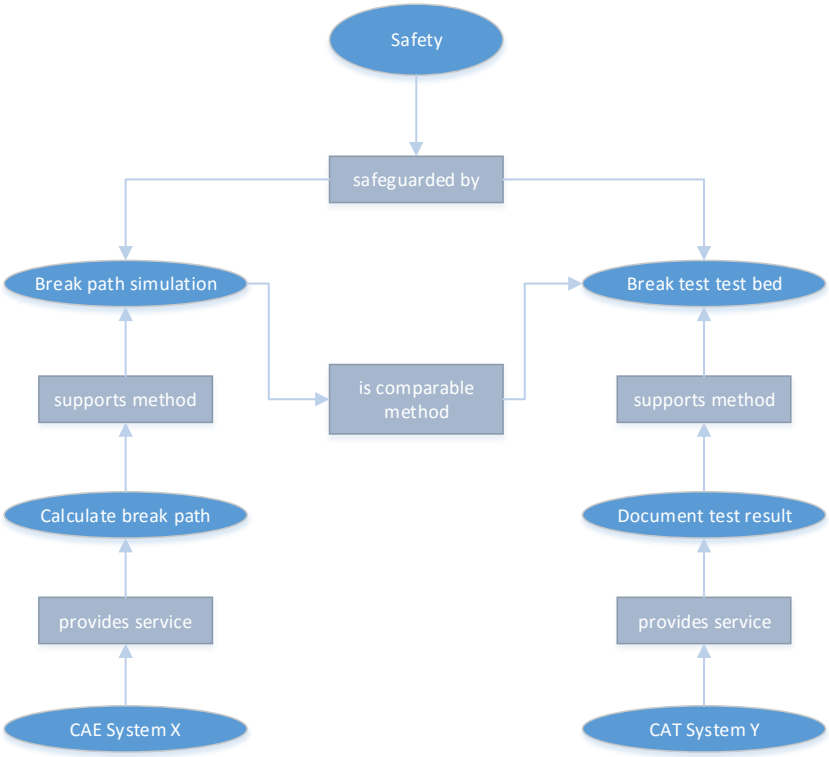


Figure 3.9.: Visualization of a CAx architecture ontology. According to C. Hess, Lambertz, and Syldatke (2009)

MEMO is a different approach for enterprise modeling and as opposed to TOGAF, MEMO provides tools, e.g., a meta model editor, that supports the users to create their customized meta models and consequently models by using standards similar to UML (MEMO OML). Furthermore, in MEMO, users use their own models, meta models, tools and DSLs. We decided to use TOGAF, though, because it is the more popular and widespread EA modeling solution and together with ArchiMate, offers the answers and meta models we need for our approach. A more detailed demarcation between the approaches is given in (Frank 2014).

Enterprise Architecture Management Pattern Catalog The final approach for EA integration we want to introduce is the Enterprise Architecture Management Pattern Catalog (Buckl et al. 2008). This document provides solutions regarding concerns, methodologies, viewpoints and information models as well as their inter-dependencies in an EA. Therefore, they introduce EAM patterns that relate to the mentioned solutions and can be applied and combined as required and

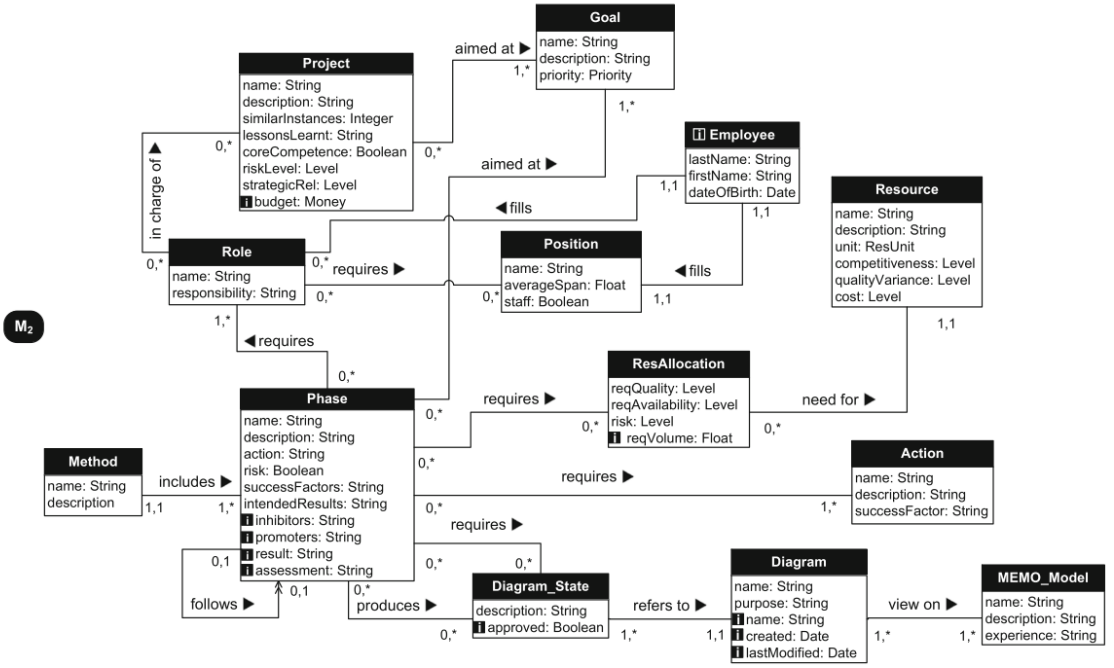


Figure 3.10.: Excerpt of a method meta model for method engineering (Frank 2014).

necessary. These patterns are divided into Methodology Patterns (*M-Pattern*), Viewpoint Patterns (*V-Pattern*) and Information Model Patterns (*I-Pattern*). As a result, these patterns can be utilized to customize, extend and supplement existing EA meta models. Since this approach is highly modular, it is possible to use ontology languages as an *I-Pattern* and provide a suitable *M-Pattern* in order to create *V-Patterns* for our relevant stakeholders, such as Domain Experts (DEs) or managers. In contrast to TOGAF, the pattern-approach is more concrete and therefore states to be implementable in a more simple way, based on best practices.

This thesis is not utilizing the pattern-approach, because TOGAF has been the obvious choice, since it is the most popular and widespread EAF. Besides, SWTs offer a very high degree of modularity for models, i.e., we have a separation between domain knowledge, business logic and business rules and also a separation inside these categories, when needed, e.g., different ontologies that can be mapped or matched. Our views are defined by the data queried by SPARQL or another query language, such as ObjectLogic, and we list method artifacts that bundle viewpoints on a business-level. A methodology for the creation of such architectures is given in this thesis and the methodology for the use, integration and maintenance of the EAM is given in ADM.

3.6. Views

In our work, we have integrated views and viewpoints for the corresponding roles. However, the creation of views is based on state-of-the-art techniques and we will elaborate the academical and best-practice foundations in the considered section. Therefore, we apply common SWT-based approaches, such as the modularization of ontologies, their mapping and of course the selection of relevant information using query languages.
However, there is of course room for improvement, as described in the remainder of this subsection.
Li et al. (2005) propose different roles with different privileges to build large-scale ontologies. The more privileges the role has, the less the magnitude of the developers that act as this role. They divide them into five categories: KMGR (Knowledge Manager), KEXP (Knowledge Expert), KENG (Knowledge Engineer), KPRO (Knowledge Proposer) and KUSR (Knowledge User) which have less and less privileges for managing an ontology, i.e. performing Create, Read, Update, Delete (CRUD) operations as well as validating the model, read from left to right.
Such an approach, when implemented, can increase the security by considering the *principle of minimal authority* (Denning 1976), offers a better usability to the respective roles and eases ontology development in general. In our case, these roles can be mapped to the mentioned stakeholders Knowledge Engineer (KE), Manager (MGR) and DE, whereas an IT expert represents a particular DE in this case (*cf.* section 6.3). A fine granular distinction with five different roles is not necessary in our case, though.

3.7. Method Selection, Analysis and Application

In the prior related work section 3.2 we have already introduced various approaches that are relevant for our work, *viz.*, the method frameworks and meta models. In order to split this thesis' objectives and hence facilitate readability, we will deal with the selection, analyses and application of these frameworks here, though, and refer to the former introduction when necessary.
Design method selection is still a topic where a lot of research is required with open research questions, like (Reich 2010):

- "How can we determine the goodness of decision-making methods in design?"
- "New issues/criteria that must be addressed when dealing with selection methods."
- "Design and validation of benchmark design scenarios that could serve to test selection methods in different contexts."
- "Application of selection methods on benchmark problems and ways to interpret the resulting data."

Nevertheless, the discipline of method selection has already been a topic in academics and enterprises for a long time. For example, having multiple methods available for the conduction of a task has already been a topic in CommonKADS (Schreiber, Akkermans, et al. 2000) that was popular in the early 90s. We do not want to walk down memory lane too far, though. Therefore, we limit our related work to the more recent past, i.e., approximately the last decade, ordered chronologically.

Method selection according to López-Mesa López-Mesa (2003) introduces many approaches for the selection of methods. As opposed to our approach, though, her work is not concerned with modeling or possible queries against that model, but it is about the algorithms for the appropriate selections. However, she proposes decision criteria which we apply in our thesis, as well (*cf.* section 6.4).

Methodos Another approach is Methodos (Franke, S. Löffler, and Deimel 2003), a method model kit that features a dialog system for the selection of methods, introduced earlier. The preselection approach they describe uses a multiple choice question system which answers are matched to methods attributes and their boundary conditions.
As illustrated in Figure 3.11 they use matrices to preselect methods that are suitable for a task at hand. In a second step, they utilize method attributes and boundary conditions to further minimize amount of results.
In contrast to our approach, they use a rigid SQL structure as their technological foundation. Besides, their focus is limited to pure design methods, because their selection process requires the input of physical parameters and effects in order to offer a proper solution for engineers. Furthermore, they do neither provide a

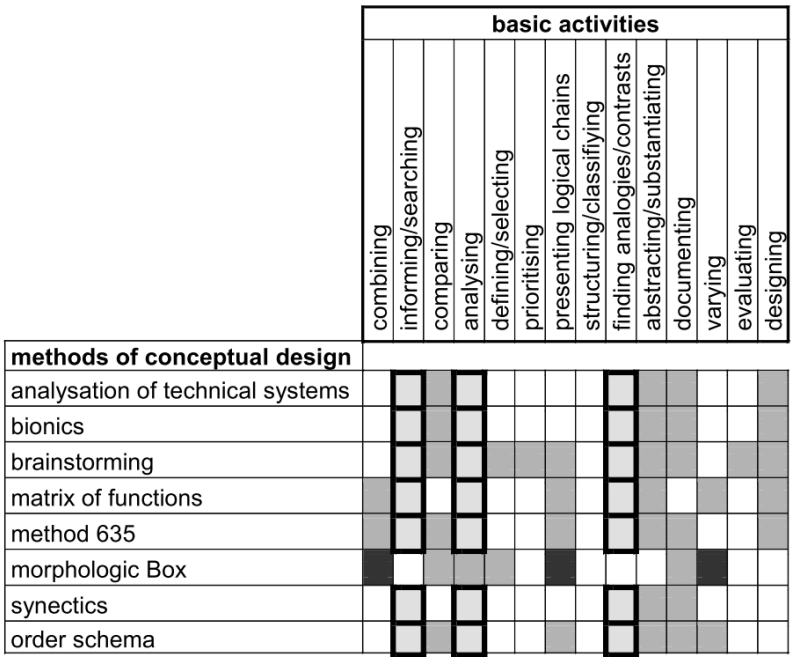


Figure 3.11.: Preselecting conceptual design methods after Franke, S. Löffler, and Deimel (2003).

methodology nor anything like an EA integration.
However, keeping in mind, that it has been published in 2003, this work is a sophisticated Internet-based, user-centric approach.

MMM A further significant contribution to the field of methods and method selection is the MMM (T. Braun and Lindemann 2003). Additionally, T. Braun and Lindemann (2004) have considered and implemented "soft" query criteria in order to select and subsequently adapt "almost fitting" methods. Principally, we ignore soft query parameters and method adaption. Albeit, that does not mean that our model is too rigid. We simply design our rules and queries to retrieve and consider only hard values.
Such an approach can be very helpful, though, when the stakeholder does not really know which method is suitable for a specific task and it is surely possible to implement a similar algorithm into our model in the future.

Methods in environmentally friendly product design and development As a matter of course, the structuring and following selection of methods concerns not only generic PDPs or PCPs but many other domains and sub-domains, such as

the work presented in Poulikidou (2012). She provides a literature review about methods and tools that are used for an environmentally friendly product design and development. Besides building on a customized domain-specific PDP, other domains also feature further and particular requirements concerning the method selection and decision criteria and making in general, e.g., "trade-offs between different environmental aspects" (ibid.) or "trade-offs between environmental, technical and quality aspects" (ibid.). Furthermore, she identified the methods' relevance to the vehicle design context.
We present this work, because it exemplifies that methods are used in a wide field of applications and therefore, an abstract method framework, like the one presented in this thesis, is a suitable basis for designing customized meta models, i.e., using our method ontology as an upper ontology. A wide array of her introduced method attributed can be rediscovered in our meta model and we show how to add further metrics, classifications, method attributes and selection criteria.

Method model kits according to Ehrlenspiel and Meerkamm Ehrlenspiel and Meerkamm (2013, Chap. 7) explain the foundations of method model kits, i.e., method frameworks or meta models and their attributes. Thereby, they define the minimum requirements a method model kit has to fulfill:

- Link between a task (work situation) and appropriate methods (method functions).
- Methods have to be described identifiable.
- Selection criteria and hints for the method application.
- Hints about further, more detailed information about individual methods or guidelines to learn them.
- A method model kit has to be extendable and updatable.

We will consider these requirements in our approach and the comparison with the rest of the related work.
Furthermore, they itemize various selection criteria, for instance, the method's accuracy and reliability, the availability of input resources, required time consumption and effort for the method execution and preparation, the business appropriateness, personal qualifications, e.g., user skill and experience, and the available aids and tools.

Moreover, Ehrlenspiel and Meerkamm demonstrate a sample of the selection criteria with 13 methods for the "determination of retention forces with snap connection", for instance, various 2D and 3D Finite Element Methods (FEMs), with the result that a method's successful and efficient application is depended on many attributes, e.g., the tested object itself and of course its examined properties (*here:* retention force). Besides, the selected example illustrates the vast amount of available methods for even very particular scopes of application which in return backs up and endorses the introduction of method meta models and kits.

Method selection after Lindström et al. Lindström et al. (2013) propose criteria for the selection of design methods on strategic and operational levels in functional product development. They demarcate between methods that can be conducted in-house or by partners due to requirements in skill and knowledge. These should be identified.
They also list and describe criteria for the selection of design methods in functional product development on the strategic level, e.g., customer requirements, organizational size/culture, competences and skills, leadership abilities, work legislation and other factors, like the state-of-art, possible required recruitment, if the methods are combinable/compatible and the method transparency.
Furthermore, they have compiled the criteria for method selection on the operational level which is the problem type, common sense, flexibility during development, the quality assurance, the level of documentation, the legal requirements, traceability, problem solving speed, the cost effectiveness and the collaborativeness/communicativeness. Additionally, they propose further factors in their related work that can be of value for any enterprise.
All these factors on both levels are of importance for generic PDPs as well and hence can and should be adapted as seen fit. Therefore, we can utilize our metrics ontology. However, the demarcation between `Concrete Methods` and `Abstract Methods` has to be regarded in some cases, because possible legislation issues may only apply to certain `Processes` and `Process Actions`.

Method selection in sustainable design after Buchert et al. In section 3.2, we have already introduced the selection and combination of methods for sustainable design in a PCP (Buchert et al. 2014), which is technologically resting upon a database approach. Therefore, we will not describe their work again, but

stress the fact, that they conclude by the results of their case study that structured method selection frameworks "have the potential to enable design engineers to select and combine methods" (ibid.) in their examined domain which is another supportive argument for our approach, as well.

Method selection according to Albers et al. Two more recent works about method selection are Albers, Reiß, Bursac, Urbanec, et al. (2014) and Albers, Reiß, Bursac, Schwarz, et al. (2015). The first paper is about the situation-appropriate method selection in a PDP. They design a mobile app that should promote the method application, because methods are applied too seldom which is due to the fact, that potential users do not know how and when to apply a method. This does not concern this thesis directly, however, it is another argument for the structured deployment of method knowledge. The kind of presentation (*here:* mobile app) has not been a focus of this thesis. However, offering this knowledge and selection possibilities on various devices is certainly useful.
In Albers, Reiß, Bursac, Schwarz, et al. (ibid.), they demonstrate, that the Integrated Product-Engineering Model (iPeM) (A. Braun, Ebel, and Albers 2013) is a well suited framework for the allocation of methods, which allows matching activities of product engineering with different activities of problem solving, i.e., methods. Furthermore, this paper explains the state-of-the-art in KM, method application and PDPs and analyses related work. Summarized, they state that "many through research developed methods are not used in practice". Therefore, KM should be encouraged and better KM methodologies need to be developed. This thesis is such an approach which can help to optimize this situation.

3.8. Comparative Assessment

In this section, we compare our approach with the previously introduced related work in general – the individual varieties have already been highlighted at the appropriate positions. The related work has been divided into the categories method frameworks/meta models, method metrics, modeling standards, EA integration, views and method selection/application. Some of them more exhaustively than others as explained below and likewise, we will treat these research fields in different ways in this section.

Because we do not compete with established maturity or metrics standards and frameworks, but want to integrate and make use of them, a comparison with our approach in that regard is not meaningful. The same applies to the modeling standards. We utilize established standards like ASAM ODS or STandard for the Exchange of Product model data (STEP) but do not offer an alternative standardized solution. Furthermore, we only briefly introduced views in order to express the possibility of extensions and improvements. The fundamental models and theories will be introduced in the relevant parts: This research field is large and we limited their application to the state-of-the-art presented in our approach and section 6.3.
Instead, we want to compare the presented method frameworks and portals with our method framework and, as a matter of fact, the fundamental meta models. Besides, the integration of knowledge into an EA will be regarded. The approaches vary in their intended target audience, their purpose and of course their technological background which is to some extent attributable to their time of publication.
While the major purpose and target audience of our approach and the other frameworks are similar and exhibits commonalities, the main difference is surely the realization of our models with SWTs. Therefore, we list the technological benefits of the SW Stack approach:

- Define and work with semantic entities that have a different syntactical representation, as in a CV.
- Every entity has a unique Internationalized Resource Identifier (IRI).
- Add meta information to every entity, like *author*, *date* or `rdfs:comment`.
- Check the consistency of the model and infer new knowledge using available reasoners.
- Querying the ontologies with SPARQL separates business logic from source code.
- Domain knowledge is separated from business knowledge.
- New and existing knowledge can be integrated using SWTs (*cf.* section 4.5).
- Use rules to express complex statements, i.e., partition business logic and code even better.

It allows a simple extension and specialization to the company's requirements, thus acting as upper ontologies. The remaining frameworks appear to have a rigid structure. Hence, their contents can be filled with new knowledge, but it

is hard to implement new classes, categories, process phases and attributes or to connect their models to a completely new domain, for instance, method metrics. Besides, the ontologies allow creating method chains and alliances using suitable and selected relations while the presented meta models usually feature just a simple hierarchy. An conceptual exception is the MMM and, when it is implemented, the PoMM, which explicitly supports the chaining with method building blocks as depicted in table 3.1. In contrast to their approaches, though, this thesis' approach has been implemented and executed by applying rules and queries in order to calculate method chains and alliances.

Meta Models / Frameworks	Framework Features															
	Formalization	Unique identifier	Extendability[a]	Updatability	Reuse legacy KBs	Method relationships	Classification	Method Selection	Method Chains	Method Alliances	Method Adaption[b]	UI	Views	Model Data	CV	Validate&Verify
This thesis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓	✓
PoMM				✓		✓	✓	✓	(✓)					✓	(✓)[c]	
MMM				✓				✓	✓	?	✓			?		
Methodos	✓		✓	✓			✓	✓				✓		✓		
MAP-Tool	✓			✓				✓				✓		✓		
PCP	✓			?				✓				?		✓		

Table 3.1.: Comparison of meta model features.

[a] of the meta model. [b] soft query criteria. [c] supports keywords.

The comparison criteria in table 3.1 illustrate various unique characteristics of this thesis' approach: (possibly global) *unique identifiers*, the *reuse* of legacy data by matching or integrating existing databases or any other KBs, the provision of *views* for different *user roles*, the *validation and verification* of (meta) model elements by using integrity rules and ontological features, for instance, consistency checks. Furthermore, the calculation of *method chains* and *alliances* is something unique, however, other frameworks support and identify such constructs conceptually in their meta models.

Besides, the table shows, that other approaches have been *formalized* as well – these solutions utilize regular database technology, though, and therefore, their extendability is limited. Only Methodos offers a concept in order to add additional meta model elements inside its UI.

Apart from providing just another method meta model, independently of the previously mentioned and following advantages, we show how to *link and reuse* existing enterprise data, for instance, databases or other ontologies. The possibility to reuse available knowledge certainly facilitates the introduction of such a method framework. Vice versa, the company's own knowledge can be easily shared with others. For example, documents that either describe a method formally (the `Procedure`) or documents that are the source or justification of a method (regulation, group policy, ...) can be deposited. That is a big advantage, due to the technological background, because stakeholders can selected methods solely based on their type of source, e.g., lawyers can monitor obligatory methods and their implementation in the PDP.
Another item, where our technology surely excels is the *classification* of methods and their *interrelationships*. We predefine some useful relations in our meta method model and the inherent relations of Web Ontology Language (OWL) offer a powerful classification mechanism, as well. Furthermore, the relations between methods and their classification can incorporate any modeled entity and concept, because this information can be taken into account easily by queries and rules.
Method selection is a feature that every evaluated related work offers in some way. Methodos, for instance, features a dialog system in order to select the right method. Other frameworks have considered the method selection in a more conceptual way (PoMM, MMM). Nevertheless, due to our architecture, i.e., the separation of logic, code and domain knowledge, we can design and create new selection algorithms fast and simple as opposed to most of the other solutions.
As mentioned, the *method adaption* is something unique to the MMM, but this thesis' approach can be easily extended to support soft query criteria.
The features our approach is totally lacking, is a developed and matured *UI*, although, one is presented in the prototypical implementations of the case studies. Further, we lack the provision of *model data*, i.e., real method data to fill our meta model. This circumstance is on purpose, though.
Finally, next to the already mentioned checks, our approach utilizes concepts like Simple Knowledge Organization System (SKOS) in order to offer a CV – a feature which enables and improves (inter)communication in the enterprise.

A further distinction between our meta model and the other approaches is that we formalize each concept connected to a method, e.g., input and output resources, tools, processes etc., as depicted in table 3.2. Thus, our model can be queried in a very fine-granular way. Furthermore, we distinguish between ab-

stract methods M_a, resources and their concrete implementation M_c and appearance in a business process, which allows monitoring, altering and analyzing every single method application.
The approach introduced in this thesis is independent of the business process, e.g., the PDP or PCP, hence, it can be integrated by simply matching the appropriate `Process` or `Process Action` concepts.
We already mentioned the capability to extend our method model with other ontologies and also illustrate how this can be achieved by introducing ontologies concerning method maturity, metrics and processes. In addition, we design a STEP-inspired resource ontology that is mappable to the method ontology. Naturally, other ontologies from the SW can be linked as well.

	Meta Model Elements														
Meta Models / Frameworks	Process	Task[a]	Tool/Aids	Resources[b]	Procedure	Purpose/Goal	Metrics	M_c/M_a[c]	Description[d]	Actor	User Role	User Context[e]	Organization	Infrastructure	Services
This thesis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓[f]	✓[f]	✓[f]	✓[f]	✓[f]	✓[f]
PoMM	✓	✓	✓	✓	✓	✓			✓			✓		✓	
MMM	✓	✓	✓	✓	✓	✓			✓						
Methodos	(✓)[g]		✓			✓			✓						
MAP-Tool	(✓)[g]					✓			✓			✓			
PCP	(✓)[g]					✓			✓		✓				

Table 3.2.: Comparison of meta model elements.

[a] or process action, work situation (*cf.* section 4.2.1). [b] in/-output.
[c] concrete (M_c) vs. abstract (M_a) methods (*cf.* section 4.3). [d] or documentation, hints, guidelines, further links.
[e] user skills, experiences etc. [f] by integrating EA. [g] one rigid process.

Finally, we integrate our meta model into an EA which is a unique characteristic to the best of our knowledge, apart from the spadework done by Syldatke, Hess et al. as exemplified in this chapter. Thus, we mutually benefit from both worlds and KBs, consisting of concepts like *Actor*, *User Role*, *Organization*, Technological Component (*Infrastructure*), Business and IT *Services* or the rest of the whole IT, data, application, business, organizational and strategical layers, as elaborated and presented in the course of this thesis.

Part II.

Method Meta Model, Method Ontology, Methodology and Enterprise Architecture Integration

"Method is much, technique is much, but inspiration is even more."

Benjamin N. Cardozo (1870 – 1938)

4

Method Meta Model and Ontology

4.1. Introduction

The task of Research & Development (R&D) business divisions comprises forming ideas for new products, i.e., the product planning, and then designing it in different stages. First, on a conceptual level, then the embodiment design and finally the detail design which is the starting point for the Start of Production (SOP) in the production division. Naturally, a multitude of different departments have to collaborate, cooperate and interact during this process which entails the application of a plethora of various technologies and methods, for instance, Computer Aided x (CAx) methods.

Our goal is to create an integrated model, realized with Semantic Web Technologies (SWTs), such as ontologies, rules and queries, that depicts this domain of methods, CAx technologies and tools in order to support Domain Experts (DEs), Knowledge Engineers (KEs) and managers in their daily work.

Until today, a method selection for a defined purpose, an analysis of the entirety of methods in a company and their monitoring have been mainly manual tasks, ever and anon assisted by basic IT documents, like spreadsheets but "impersonal transfer of product development method know-how by method-databases or further multimedia approaches is gaining in importance" (T. Braun and Lindemann

2003). "Of course, the individual coaching of methods by trainers or consultants is superior to impersonal forms of method transfer, like literature, web-based method libraries or databases. Nevertheless method implementation and application by consultants or trainers is often limited to a specific existing repertoire of methods. This is the reason why methods often are applied that do not solve the actual problem" (T. Braun and Lindemann 2003).
Birkhofer described this circumstance in the domain of Design Theory and Methodology (DTM) as follows: "In reality, there is a wide variety of proven and effective design methods and CAx tools available. An engineer can choose the method or tool most appropriate to the task and stage of the [Product Development Process (PDP)]" (Birkhofer 2011). However, in view of the fact of the vast amount of available methods it has been almost impossible to keep track of this "method jungle" for the potential user (Stefan Löffler and Jagusch 2004) and it still is. Furthermore, the knowledge about these methods is scarcely formally documented. This hinders a comprehensive analysis and impedes the planning and monitoring of the working methods.
Using methods in the PDP is reasonable, because human cognitive abilities are limited (Lindemann 2009). Methods represent a tool for developers that allows handling the complexity of the various facts and circumstances. With the assistance of working methods, complex problems can be broken down into manageable partial problems. Conflicting goals can be detected and they help to identify the appropriate field of attention. Methods can even help to overcome thought barriers and encouraging the required creativity for the PDP (ibid.). That is, the application of methods does not only have advantages – critical aspects have to be regarded as well (ibid.): It comes along with a certain effort.
The goal of the task at hand should be precisely addressed at the beginning of each method application that supports achieving this objective (T. Braun 2005). Therefore, the users have to reflect about each method application in retrospect and thereby gather experiences for future applications (Lindemann 2002).
In the following, we will define the term method, because it is ambiguously used in many disciplines. In this context, we delineate the term by comparing it with and relating it to processes, tools, systems and methodological definitions. In order to understand, compare and assess these diverse methods, and therefore enable their interaction, we create a base method model, realized with SWTs. Afterwards, we are extending the base method model piecemeal by considering existing work and standards. This model describes the methods' contexts, e.g.,

the tools and people that use these methods, their inter-relationships, attributes, maturity and so on. It does not explain how a single method is to be used in detail – this information can be deposited – but acts as the basis of a framework for DEs, such as engineers. More precise, the methods are considered as a black box in our method framework, i.e., we know its input and output but its implementation is opaque, although we have a rough understanding of its transfer function by interpreting its semantic context information and annotated description.
For example, when performing a physical vehicle crash test, a bunch of input items are required for the test setup, e.g., a testing facility, crash test dummies, cameras, barriers and of course the to be tested prototype. Besides, a lot of people are involved in such a test, for instance, technical staff, mechanics and engineers. Please note, that we are not planning to provide a model that is suitable for a detailed Enterprise Resource Planning (ERP). In the provided example, it can be assumed that the availability of the to be tested prototype is a sufficient information for a rough-granular method planning – this information alone allows isolating the relevant business tasks along with its associated process knowledge. The expected output may be, depending on the particular kind of crash test method, raw data for the calculations of the mechanical stress or forces that take effect on the crash test dummies such as the Head Injury Criterion (HIC) or even more abstract, e.g., "sufficiently safe". Again, the level of detail required for our purpose is an abstract one, hence the final output "HIC" may be a satisfactory information but it depends on the demanded analysis. Besides, the details about the crash test itself, the preparations, the actual setup, the execution and so on are also not part of our framework.
Greiffenberg defines the term framework, which is well known from software development, for the domain of methods (Greiffenberg 2003). Originally, it is an approach for reusing software in a specific problem domain, offering the developer an infrastructure for possible applications. A framework therefore provides a reference architecture and implementation for a family of software systems. The term framework can easily be transferred to the domain of methods, acting as a method framework that provides an architecture for the method landscape in an enterprise.
This framework's purpose is the selection of the best fitting method(s) for a predefined objective while considering functional and non-functional requirements, for example, business rules (BRs), estimated costs, quality or time consumption. By defining the Key Performance Indicators (KPIs) using rules and queries, they are

strictly separated from the software application, which interacts with the model and user. The DEs can customize selection criteria and adjust and extend the model according to their needs on their own or together with a KE.
For example, instead of a physical driveability test on a test route, parts of the vehicle's performance can be simulated virtually. Both methods, resulting in an assurance of or a statement about vehicle properties, markedly differ in financial expense, the quality of the result and its duration, including preparations and follow-up procedures.
In this context, it is important to consider the method's environment and the requirements regarding its application, as listed in Table 4.1.

Possible Questions about the method environment
What kind of resources are required to apply the method (input)?
Which resources are produced (output)?
Who is conducting the method?
Which role is best suited?
Which goal is pursued?
In which process can an arbitrary method be implemented?
When can a specific method be conducted in a given business process?
Are there any scenarios where its implementation is not recommended?
Are all requirements for executing the method fulfilled?
When are the results required?
How long does it take to conduct the method?
What is the expected quality of the result?
How can methods be compared?
Which method (combination) is best suited for a task at hand, considering attributes like duration, quality and costs?
How does the method work?
How can I conduct the method?
Which tools are needed in order to conduct the method?
"Which CAx systems are implementing the required CAx methods?" (C. Hess, Chen, and Syldatke 2008)
"How can one exchange information between specific CAx systems?" (ibid.)

Table 4.1.: Exemplary questions regarding a method application and environment.

Our model shall be able to answer these questions and thereby support the DEs to choose the best fitting method for the scenario at hand. And vice versa, lacunae and blank spots, i.e., missing entries in the Knowledge Base (KB) or even real absence of required methods, can be detected which helps in the strategic planning of the business. Drawbacks and advantages of methods existing in the company can be compared and therefore a strategic reuse, a tailoring, the adaption or the development of completely new methods can be launched.
Furthermore, the model can be used in order to query for *methods chains* when one method is not sufficient for achieving the expected goal. These sets of methods are particularly demanded by Small and medium enterprises (SMEs) in order to handle entire process actions, because, according to T. Braun and Lindemann, SMEs often refuse to work with the selection of single methods blocks (T. Braun and Lindemann 2004). These *method chains* can be realized by aligning the interim results by considering the methods' predefined inputs and outputs in the KB.
Another benefit of a formalized method KB is a standardized and mutually accepted description and vocabulary in the enterprise. By connecting the methods to other business objects, they can be found with context information, even if their specific name is unknown for the DE. By sharing the method knowledge, unnecessary redevelopments and therefore also unnecessary expenses can be prevented. SWTs also support alternative labels for the sundry entities which might be used for a multilingual application or for different terms representing the same concept in the business divisions. Using the advantages of reasoning with ontologies and rules, inconsistencies in the KB can be easily detected and new knowledge can be inferred.
Altogether, the application of methods leads to a minimization of risks during the product development, i.e., reaching the set goals without hitting the emergence of crises (Lindemann 2009).

4.2. Method Definition

"The concept of method has been discussed for several decades" (Cronholm and Ågerfalk 1999) in the past, is still discussed, is used in an ambiguous way in various literature sources and is also not clearly defined in enterprise vocabulary. Especially when considering that the term is used in many disciplines, i.e., scientific, design or programming methods as illustrated in Table 4.2.

Coarse Distinction	Fine Distinction	Examples (technical and others)
1. Area of application "human"	1.1 Individuals	Problem solution, procedure cycle; procedure plan; design rules; calculation methods, to-do lists
	1.2 Groups, Enterprise: interpersonal	Procedure plan; project management; FMEA; QFD; methods of work preparation; methods for enterprise, state organization; economical methods
2. Type of application by the "human"	2.1 Tacit, unwittingly	Intuitive problem solving; designing; constructing; trained behaviors for foreign languages; communication; driving a car; handling a computer. Even cycling or swimming.
	2.2 Wittingly, rationally communicable	Calculation methods; organization methods; teaching and training methods for production, assembling, methods for documentation, searching; design methods
3. Abstraction level	3.1 Abstract	Function analysis and synthesis; mathematical methods; language grammar; natural science, philosophical analysis methods
	3.2 Concrete	Teaching and training methods for production, assembling; swimming, skiing, for the implementation of internal organizational processes; medical treatment and surgical methods.
4. Area of application	4.1 Humane Sciences	Philosophical analysis methods; methods of literature analysis, exegesis, musical analysis, legal methods
	4.2 Natural Sciences, Mathematics etc.	Proof, calculation methods; methods of computer science, physics, chemistry, biology; medical treatments; weather forecast
	4.3 Engineering	Calculation methods for solidity etc.; FEM, methods of measurements, CAD, VR, AR, rapid prototyping methods; production, assembly methods; construction, design, sales methods
5. Goal	5.1 Analysis of properties	Calculation methods for product properties; methods of measurements; quality assurance methods; simulation methods; statistical methods; cost accounting methods; balance sheet analysis
	5.2 Synthesis	Design methodology with an emphasis on function; resp. property X; methodology of project, plant, construction and architecture planning; composition of music

Table 4.2.: Overview and classification of exemplary methods and tools. According to Ehrlenspiel and Meerkamm (2013, trans.).

Depending on the department or company, even of the same domain, e.g., automotive engineering, a method can be synonymous to a tool, a process, a kind of technique for solving or analyzing problems or a combination thereof. In software engineering, the term is usually associated with a procedure, i.e., a segment of a framework, of a software development methodology, e.g., the waterfall model or an agile development framework.

Besides, people tend to use vagueness when talking about concepts (Fischer 2013, p. 28): depending on the organization and its business sector, a specialization of a method, e.g., a programming method in an Information Technology (IT) com-

pany, or a Computer Aided Engineering (CAE) method in tool construction, is just called method, therefore the word is underspecified. This ambiguity impedes the communication between people with a different background and leads to unnecessary misunderstandings and coordination phases.
Thus, a standardization of the semantics would be worthwhile. To achieve a common understanding, we created a method meta model that has been implemented with Web Ontology Language (OWL) later on, by working closely together with industrial partners and regarding state-of-the-art definitions and models. Further necessary knowledge that has been derived from our gathered requirements, i.e., to answer requested information, has been implemented in the form of queries and rules.

4.2.1. Definition

The term "*method*" originates "via Latin from Greek methodos 'pursuit of knowledge', from meta- (expressing development) + hodos 'way'" (Oxford Dictionaries 2015). A method is "a particular procedure for accomplishing or approaching something, especially a systematic or established one", e.g., "a method for software maintenance" or "a labour-intensive production method" (ibid.).
WordNet (2013a) describes a method as "a way of doing something, especially a systematic way; implies an orderly logical arrangement (usually in steps)". Merriam-Webster.com (2015) defines a method as "a procedure or process for attaining an object", i.e., "a systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art" and "a way, technique, or process of or for doing something" (ibid.).
According to Duden (2015), a method is defined as a procedure, based on a rule system, for the attainment of [scientific] insights or practical results.
All these definitions have in common, that a method is a systematic procedure for the attainment of something. Nevertheless, they are occasionally mixing up processes, techniques and procedures and in turn are using the term method *idem per idem* to explain these terms, e.g., *technique*: "a practical method [...] applied to some particular task" (WordNet 2013b).
A more precise distinction of procedures and methods is formulated by Hesse, Merbeth, and Frölich (1992): "procedures are executable regulations or instructions for the well-aimed application of methods. A method can be supported by several (alternative or mutually complementary) procedures."

Definition 4.1 (*Task*[1])

"A task defines some work to be done and can be specified in a number of ways, including a textual description in a file or an electronic mail message, a form, or a computer program" (Georgakopoulos, Hornick, and Sheth 1995).

A similar definition is formulated by Chroust (1992): "a procedure describes a concrete way of solving specific problems and problem classes [...]. A method can be realized by several alternative or by several aggregated procedures."
These definitions tell us that a method is not the same as a procedure, but that a method consists of one or many procedures that are interconnected by logical rules (*alternative*, *predecessor*, *successor*, etc.) (*cf.* Figure 4.6 on page 114).
Because our running example originates from the engineering domain, we also incorporate domain specific method definitions and are inspired by existing method frameworks, e.g., from systems engineering and method development (*cf.* Weigt 2008).
Methods have been used for decades in the domain of DTM (Birkhofer 2011) – an approach for the methodical development of products by using "effective methods to support particular development steps and [guide users] to efficiently solve development tasks, often with remarkable success" (ibid.). However, according to Birkhofer, the industry only reluctantly adapts design methodological models and methods (Pahl, Beitz, Feldhusen, and K. Grote 2007).
Pahl et al. define DTM as follows:

Definition 4.2 (*Design Theory and Methodology*)

"Design Methodology is understood as a concrete course of action for the design of technical systems that derives its knowledge from design science and cognitive psychology, and from practical experience in different domains. It includes plans of action that link working steps and design phases according to content and organization. These plans must be adapted in a flexible manner to the specific task at hand. It also includes strategies, rules and principles to achieve general and specific goals as well as methods to solve individual design problems or partial tasks." (Pahl, Beitz, Feldhusen, and K.-H. Grote 2007)

This definition about the methodological approach, methods and their context include many noteworthy aspects.

[1] Following the example of Eisenbarth (2013), please note that we use the terms "task" and "process action" synonymously.

Methods are prescriptive which means they are perceived as some kind of instruction or plan (Lindemann 2009). Nevertheless, plans and methods must be adapted to the specific situation they are used in which means that a method can produce entirely different results and qualities depending on its environment. Among others, this includes the user itself. They have different knowledge, experiences, skills, tendencies and their form of the day is variable as well. Furthermore, the quality *depends on the tools* that are used to execute a method, the *type of business process* a method supports and of course the *quality of its input parameters*. Because the term tool can be ambiguous as well, we are using a definition by Weigt (2008) as a basis for our definition in this thesis:

Definition 4.3 (*Tool*)

"A tool [, i.e., an aid,] for the processing of a task is something that directly supports the processing within the scope of a method, provided that at least one substantial component is required for this effect, excluding the processing agent [, e.g., the user]" (ibid., trans.).
In contrast to Weigt's definition, our tool definition also comprises insubstantial, more precisely virtual, components, e.g., a computer program.

Definition 4.4 (*Model*)

"A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language" (J. Miller and Mukerji 2003).
Furthermore, "a model only reflects a (relevant) selection of the original's properties" (Brambilla, Cabot, and Wimmer 2012).

Next to methods, the domain of Design Methodology comprises strategies, rules and principles in order to achieve some business goal. Our approach also strongly emphasizes on the use of rules in combination with KBs along with the rest of the SWTs. Consequently, we can say that our approach can be classified as a contribution to the domain of Design Methodology. Furthermore, Birkhofer stresses the fact that Design Methodology can and should be aligned to CAx technology because of the many existing parallels: "design methods and CAx tools fit together like a key and a key hole" (Birkhofer 2011). Nevertheless, according to Meerkamm (2011), both domains have to be further adapted and their interaction

and combination improved in order to gain more efficiency. We assume that the meta model and framework presented in this thesis along with the methodology and the case studies help to improve this liaison.
Especially the Property-Driven Development (PDD) use case, presented in section 7.5, demonstrates many similarities between DTM and the Characteristics-Properties Modeling (CPM), respectively PDD, approach which is a fact also surveyed by Weber (2011).

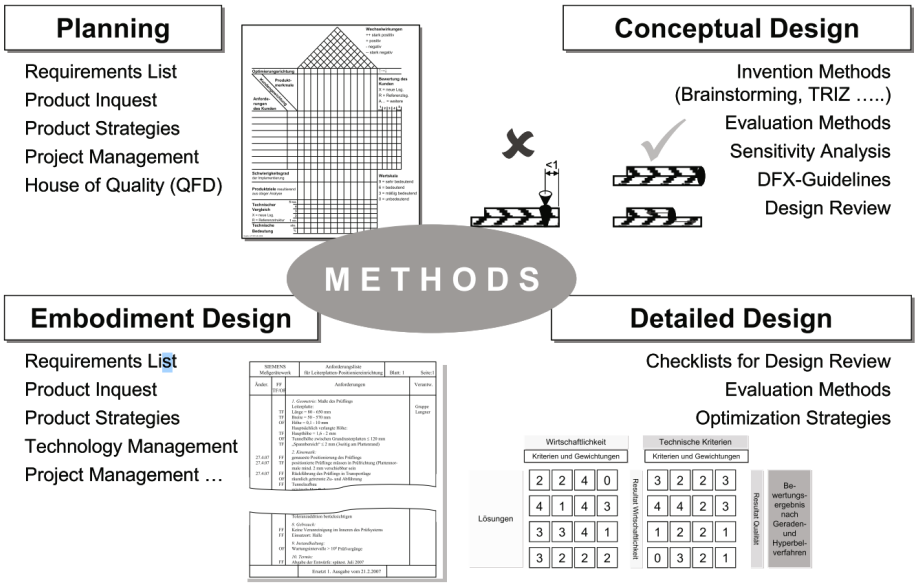


Figure 4.1.: An overview of various design methods (Meerkamm 2011).

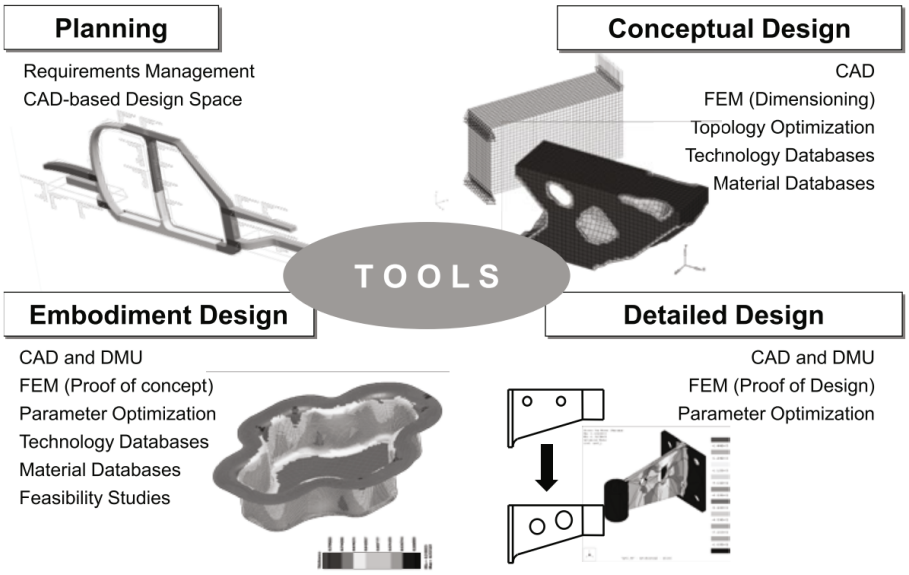


Figure 4.2.: An overview of various tools (Meerkamm 2011).

In the figures 4.1 and 4.2, Meerkamm (2011) depicted various methods and tools for the product development. The figures are partitioned into four phases of a PDP: planning, conceptual design, embodiment design and detailed design. Thus, methods cannot be applied universally, but *every method has a particular stage in the PDP* where its execution offers the best outcome. A detailed interaction description and demarcation between methods and processes is explained in the following section 4.2.2. For now, it is important that methods and tools can be classified into the different stages and into different categories inside these stages. Meerkamm mentions, that the interaction between methods and tools has to be improved in the future. Our method ontology offers a model that allows semantically expressing the relations of tools and methods in combination with their tasks and processes on an abstract layer but also offers the possibility to describe concrete methods and their boundary conditions, e.g., the available time or budget. Figure 4.2 focuses on virtual tools, i.e., tools from the Computer Aided Design (CAD) and CAE domain, whereas our model also supports Computer Aided Testing (CAT) tools and test environments, that is physical hardware, for instance, a wind tunnel or test bench. Furthermore, our model allows annotating the quality of their interactions, based on a maturity scale described in section 4.4.1.

According to Mueller (1990, p. 17), also from the domain of engineering, a method is defined as a set of instructions; when applied, it sufficiently ensures the execution of an operation sequence that is considered appropriate under given conditions. Expediently, the methodical support should be coordinated with its respective boundary conditions (ibid.). In particular, it should be differentiated between the boundary conditions of the *organizational and operational layers* (Weigt 2008). Boundary conditions of the organizational kind include *role descriptions*, the *mapping to business processes*, *organizational structures* etc. The operational layer comprises information about the *tools* used to conduct the method, its *qualities*, *time consumption* and *costs*.

Two other concepts have to be demarcated from the above semantic of the term method. The *methodology* in the sense of scientific studies about methods and method systems (Mueller 1990, p. 2) and the methodology (or methods) with the meaning of development and analysis of methods of a specific or a class of domains. Or more precise, the outcome of such an endeavor, a domain-specific system of adequate methods (ibid., p. 17) that tell us “what”, “when” and “who” should perform a task.

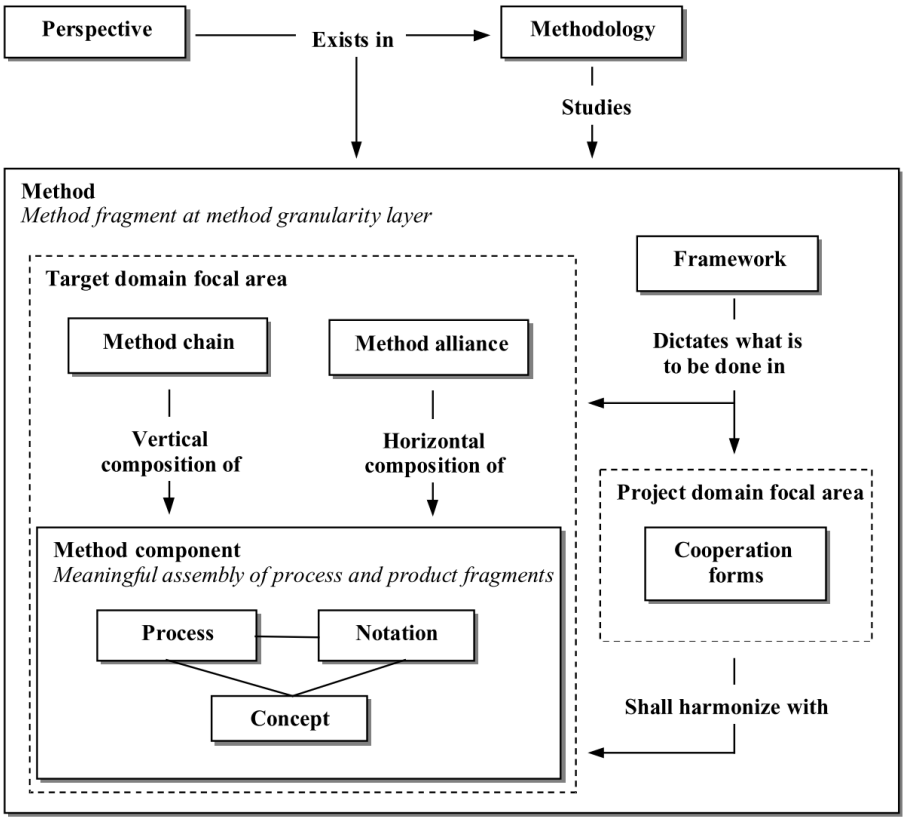


Figure 4.3.: A conceptual model of the method concept and its relations (Cronholm and Ågerfalk 1999).

Cronholm and Ågerfalk (1999) define the relations between different method concepts in even more detail (*cf.* Figure 4.3). The depicted conceptual model introduces some new important concepts: on the one hand, methods can be vertically linked, thus creating *method chains* – the outcome of the previous method is the input for the next method. In the diagram, method chains are formed by method components – for us, a method component is defined as a method again. A *method alliance* on the other hand, is a horizontal composition of methods, i.e., a family of similar, alternative methods. Cronholm and Ågerfalk also distinguish between two focal areas: the project and the target domain, which is not important for our work, because we just focus on methods of one domain, for instance, CAx methods. Nevertheless, they depict the concept *framework* which "dictates what is to be done in" the domain focal area. The final important concept is the *perspective* which resembles our views (see section 6.3), i.e., how methods are perceived and what level of detail with which focus is presented to the user.

Another vital aspect for the definition of methods, more precise for the framework of methods (ibid.), is the *methods' structure*. According to Lindemann (2009), it is not a simple task to clearly classify methods and put them in some hierarchical structure. A kind of network or graph, e.g., an ontology, is much more suited for this task, because single methods and their partial procedures can be applied in other methods as modules as well. "This modular character supports a flexible selection, adaption and combination of methods" (ibid.).

4.2.2. Methods vs. Processes

As already addressed in the previous section, the distinction between processes and methods is not straightforward. However, we need this distinction, because our goal is to combine method knowledge with business processes in the following chapters.
One of the most prominent process definitions has been formulated by Davenport, where a process is identified as "a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus's emphasis on what. A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs" (Davenport 1993).
We can find many similarities between this definition of a process and our understanding of a method. Both are identified by a structured or systematic set of activities or procedures that produce an output or attain a goal, i.e., they have a purpose. The method however, is not focused on a customer or a market, but exists just for the purpose of attaining something - the following utilization in the business is irrelevant for a method. Furthermore, both concepts emphasize on the way something is done and not on the product. The concept of time and space of a process, e.g., a report that has to be available at a specific place and milestone, is different to the concept of time and space in our method definition. In the case of a method we do not relate time and space to its environment, i.e., the organization, but time is only considered internally - the aggregate of times needed to perform the procedures contained in a method defines the time consumption of the method itself. The last part of Davenport's definition tells us, that we have clearly identified pre- and post-conditions when performing a process.

This statement also holds true for a method, when heeding the constraint that the post-condition or output may differ in quality based on the maturity and quality of the method itself. This constraint also applies to the inputs of a succeeding method, when previously generated output has further use.
According to T. Hess (1996), a process can be described as a sub-system of the business' workflow organization. Next to the workflow organization, T. Hess defines two further organization sub-systems: the organizational structure and the information systems. The process' elements comprise tasks, their associated roles and resources. The relations between those elements represent the workflow.
This definition obviously shows a lot of similarities to the method definition: our methods use resources and are executed by roles. However, the methods just represent a possible operation of a task. They explain the how and not the what. Thus, the resources and roles can just be implicitly linked to the method, because they can be defined in the process itself. Nevertheless, it can be expected that not all methods are connected to processes in the model, either because the connection is not yet known, is simply not yet modeled or the method is really not used in any process at the given moment. For example, when a new method is to be introduced in the PDP it is possible to define the connection in a target enterprise architecture but not in the baseline enterprise architecture. This means, that both options, using the connected process' roles and resources and the possibility to define own roles and resources for the method, must be facultative.
In his thesis, Weigt (2008) elucidates that methods invariably are mental procedural models of processes. This subsumption does not necessarily hold true the other way around, though (ibid.). Whether a process model is considered as a method depends on the pragmatic feature of the concrete modeling.

Definition 4.5 (*Process Model*)

"A process model is an abstract set of process actions, being arranged with a number of patterns that define the control-flow of the process actions" (Eisenbarth 2013).

A model's pragmatic feature expresses the modality of the relationship between original and model. Some features or properties of the original may not be important for a specific purpose, for instance, a process can be abstracted to just cover costs. Such an abstracted original can be regarded as a model in the domain of business administration. However, such a model does not constitute a set of instructions (*cf.* Mueller 1990, p. 17), consequently it does not act as a method.

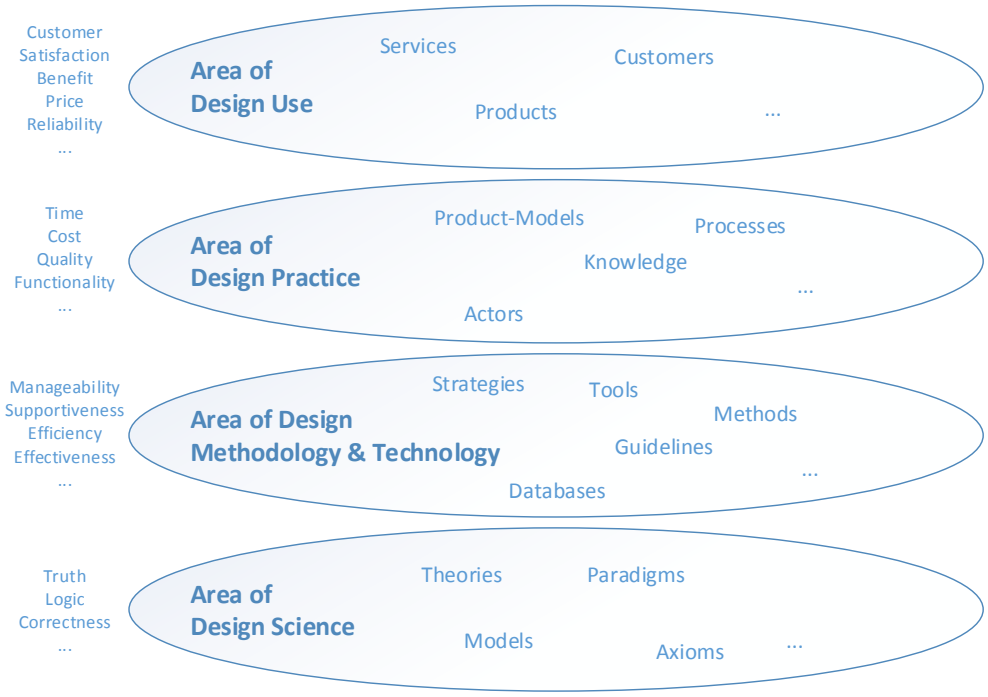


Figure 4.4.: A classification of the four design-relevant activity areas together with goals. According to Birkhofer (2011).

Hammer and Champy (1993) define a process "as a collection of activities that takes one or more kinds of input and creates an output that is of value for the customer". In our model, methods do not link to customer satisfaction.
When we classify the customer in one of the four design-relevant areas (Birkhofer 2011) (*cf.* Figure 4.4), it best fits into the area of Design Use, the top layer, which comprises products and services. The goals of this layer are customer satisfaction, reliability, an overall benefit, an adequate price etc. The next layer is the area of Design Practice. This area consists of the processes, knowledge and product-models in order to satisfy the customer. The important goals here are quality attributes like time, quality, cost and functionality. The remaining two layer, the Design Methodology and Technology area and the Design Science, together with the area of Design Practice, are the areas where our model, its application and benefits fit best. While the third layer is about conditioning and documenting knowledge gained from the upper layers by abstraction and modeling, for instance, methods and tools, the bottom layer describes theories, axioms and paradigms, e.g., by criteria such as truth, logic and correctness, of the upper layers. According to Becker and Vossen (1996), business processes also feature interfaces to external

market partners. This is clearly a characteristic of processes, but methods only feature links to the roles or people they are conducted by.
Other authors, like Österle (1995), emphasize on the use of information systems to support the execution of processes in their definition, whereas we use the more abstract concept *tool* regarding methods that can represent either a software application or a physical device or implement. However, Österle defines a task in an enterprise as an organizational function that can be conducted by humans or machines. These tasks can be aggregated to represent higher level tasks right up to possible all-encompassing tasks that represent the enterprise's core business goals. This characteristic partially applies to our methods, as well. They do not specifically aim to support business goals, but goals in general (every method has a purpose), but can also be consolidated in order to describe more complex and greater methods. Likewise, procedures can be summarized by a single method.
Because a method's application fundamentally serves the achievement of a goal, i.e., it is purposeful, they consequently comprise a *function* (Weigt 2008).

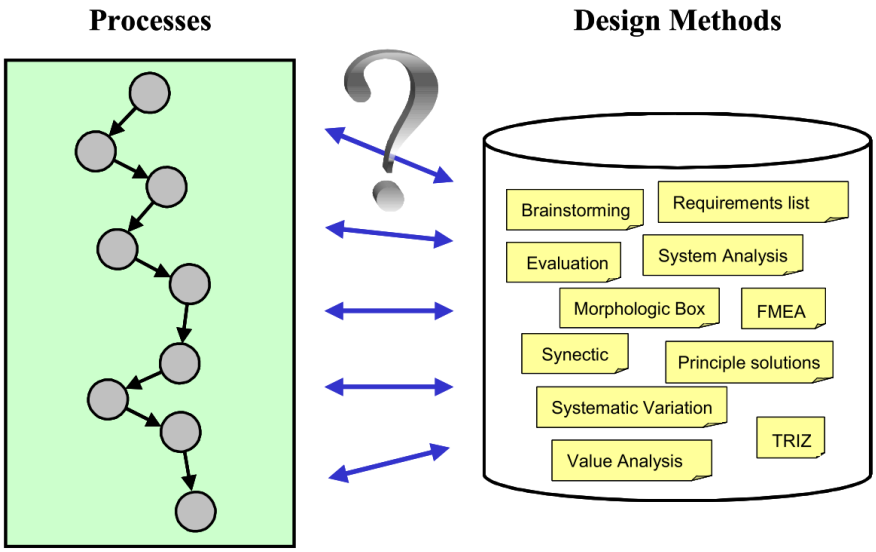


Figure 4.5.: Processes and design methods in a PDP (Birkhofer et al. 2002).

Birkhofer et al. (2002) state, that there are many tools and methods or even a mix of methods to support single sub-processes (*cf.* Figure 4.5). In his work, he introduces a method framework that should support a standardized method pool in order to support DEs in their daily work. His paper is more thoroughly analyzed in the related work chapter 3.

Now that we have illustrated the differences and similarities between processes and methods, we want to explain how they interact with each other.
Lindemann (2009) states, that methods are used in PDPs for supporting a systematic and aimed execution of tasks. He brings methods into line with the Three-Layer-Model (Giapoulis 1998), one of many models describing the PDP.

- The **layer of strategic planning** consists of abstract partial projects, phases ("clarify task", "create CAD model" etc.) and a rough time schedule. This is the layer, where the project management is conducted.
- The **layer of operative planning** comprises concrete processes that are executed in the various strategic planned phases, for instance, "check CAD model for collisions" or "correct CAD model". This is the layer where methods can be applied in order to support the PDP.
- The third layer is the **result layer**. For example, results can be requirements, measurements or CAD models.

According to Giapoulis, methodologies, basic principles and methods can be used as support and reference for the strategic as well as the operative layer. Our method model, in consensus with the above Three-Layer-Model, also features different method concepts.

4.3. The Core Method Ontology

The core method ontology's structure is depicted in Figure 4.6. We use the attribute "core", because this ontology will get extended by other ontologies in the next sections and chapters. The ellipses in the figure represent ontology classes and the directed edges stand for properties, whereas their beginning is the property's domain and the arrowhead represents its range. Multiple properties between the same classes are consolidated into a single edge for a better overview; they are separated by commas between the edges' labels, though.

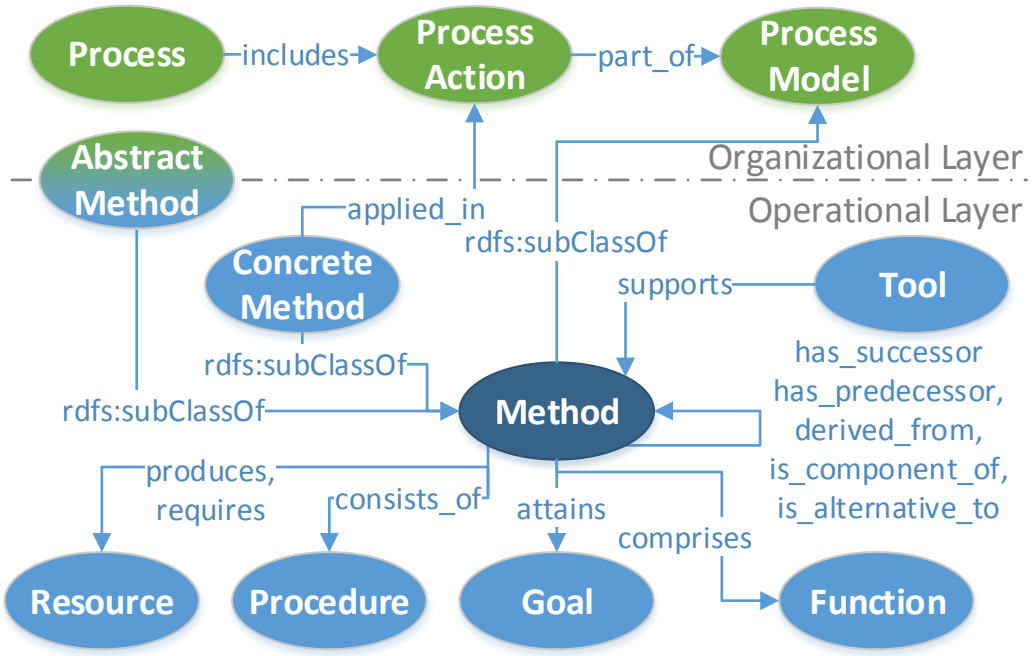


Figure 4.6.: Structure of the core method ontology.

4.3.1. The Ontology Concepts in Detail

In Figure 4.6, both, the `Abstract Method` and the `Concrete Method`[2], are specializations of the concept `Method`. The `Method` is a specialized type of `Process model` as explained before. Thus, the `Abstract Method` is an entity representing a method on the strategic planning layer. In contrast, the `Concrete Method` is the instance of such an abstract method.

Definition 4.6 (*Abstract Method*)
An abstract method M_a is a concept that defines a class of process-independent methods.

Definition 4.7 (*Concrete Method*)
A concrete method M_c is a concept that defines a class of methods, linked to a process action and derived from an abstract method M_a.

[2] A more precise distinction between the method types `Abstract Method` and `Concrete Method` is described in section 4.4.2.

Methods offer suggestions for specific tasks' sequences and the fashion on how these tasks are to be conducted (Lindemann 2009). As defined earlier, a *Process Action* is equivalent to a task and hence, a *Concrete Method* is *applied_in* a specific task that is again related to a `Process`. Because both method types inherit the object properties of `Method`, we can model inter-relationships between `Concrete Methods` and respectively `Abstract Methods`, and can also state that some arbitrary `Concrete Method` is `derived_from` a specific `Abstract Method`. The remaining object properties allow us to model classes of methods such as method chains and alliances (`has_successor`, `has_predecessor`, `is_alternative_to`). For instance, we can express that a collection of methods, i.e., a method class, can be applied to solve equal or similar problems.
The three concepts at the bottom right of Figure 4.6, namely `Procedure`, `Goal` and `Function`, are used to formally describe the methods' contexts, resp. its comprised procedures.
In accordance with the previously mentioned definitions by Hesse et al., Chroust etc. (Hesse, Merbeth, and Frölich 1992; Chroust 1992), one or many procedures is a composition that is part of a method in our ontology. Thus, the class `Procedure` is used to describe partial method steps that are not methods itself. The relation between methods and procedures is reflected by the object property *consists_of*. An instance of the class `Procedure` can either describe this procedure in textual form with the given Resource Description Framework Schema (RDFS) properties (*rdfs:comment*, *rdfs:seeAlso*, *rdfs:label*, ...) or can be linked to more complex constructs, like a class `Document` (not part of the figure) which again may link to a document in the file system. Otherwise, if a method makes use of another method, object properties, like `is_component_of` shall be used.
Taking into account, that a method is always purposeful and therefore always focused on a solution of a problem or task (Lindemann 2009), a `Method attains` the concept `Goal`. The method's `Goal` is the class that can describe the method's contribution to the enterprise's strategy or overall value creation. For example, we can create the classes `ProductAssurance` or `VehiclePropertyAssurance` as a `Goal`'s subclasses. When a specific method assures an arbitrary business product, the respective instance can be linked to the `Product` of a product ontology. This concept is further described in chapter 6.
Usually, methods are supported by tools that are to make its application more effective and efficient (ibid.). "Generally, tools have a great impact on to method's success. That way, the availability of a tool when applying a method is changing

the situation strikingly for the user, if he is experienced with its handling" (Lindemann 2009, trans.). In our ontology, this is reflected by the property `supports` between the concepts `Tool` and `Method` which means that a method can be supported by one or many tools. The term and therefore the respective class `Tool` covers a wide range of different auxiliaries or assistive equipment. In the domain of CAx, this can be all kind of testing equipment, physical implements, simulation or modeling software, but also software for ERP, statistical analyses or something totally different. Besides, the term also comprises simple assistive things, like forms, checklists etc. If required, sub-concepts, such as `Aids`, `Computer Tools` or `Hardware Tools` and others mentioned in Ehrlenspiel and Meerkamm (2013) can be introduced in order to create a more fine-grained classification.
The third layer of Giapoulis's model, the result layer, is only vaguely considered in this figure. On the one hand, the aforementioned `Goal` comprises some kind of the method's result. On the other hand, the ontology incorporates `Resources` that are produced, but also required, by a method. They are important for answering many of the questions of this chapter's introduction and therefore are introduced in more detail in the complete representation of our method model in section 4.6.

4.3.2. Representation Models of Methods

Methods can be distinguished by their kind of representation. Basically, they can be represented in four different formulations[3] (Weigt 2008), whereas only the two formulation definitions below are relevant for this thesis:

Definition 4.8 (*Flowchart*)

"Preferably a sequential order of guidelines, rules, instructions or commands applying a concerted universal set of symbols, possibly with loops and branching" (Mueller 1990, trans.).

[3] flowchart, dialog, template and principle

Definition 4.9 (*Template*)

"An arrangement of characters (names, icons, symbols) in a familiar holistic configuration (form, pattern, formula) so that the processing agent can apply and interpret them regarding a concrete use case and hence is guided, compelled or forced to use procedures for a purposeful application" (ibid., trans.).

The flowchart's purpose, depicted in Figure 4.7, is to describe the application of a method in detail in a specific scenario, whereas our approach is more abstract. It is comparable to the method description as a template, whereas we only focus on the inputs and outputs. Circumstances and by-effects are considered to be the contents of the method, regarded as a black box, in our method model. Our model represents the entirety, the system, of methods in a business domain or the whole business and supports the relevant roles, e.g., a methods engineer or a DE, in choosing the right method for their purpose.

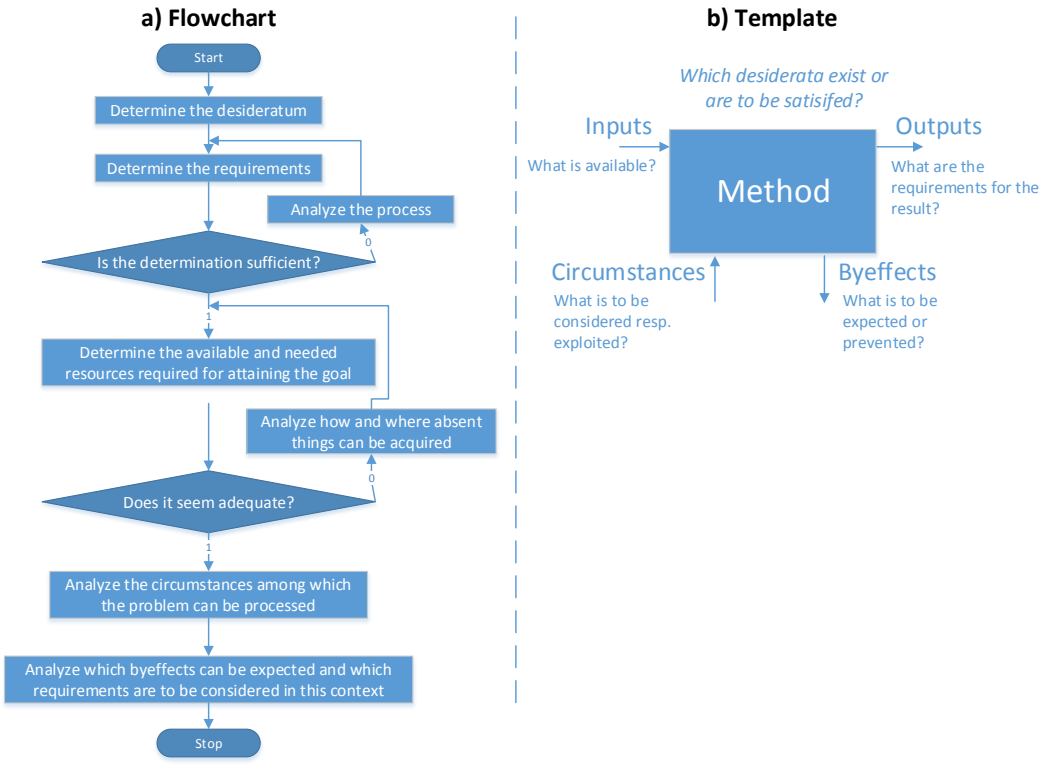


Figure 4.7.: Method formulations for "Specifying a task" as a flowchart *(a)* and template *(b)*. According to Mueller (1990, p. 25, trans.).

Since annotations can be deposited to the ontological concepts and instances or the ontology itself can even be extended, the method's detailed description can be deposited but our focus is the method's setting, influence and other meta information.
Nonetheless, the existing method formulations in a business are an important source of information for our ontologies. Existing formulations in the company state which concepts, object and data properties are important on a structural level. Different formulations that derive its origin from different departments or roles might feature a varying granularity, focus and kind of representation. They provide an indication for the kind of view that should be presented to the various user roles, depending on the current preference and task. Nonetheless, all the declarative information are to be formalized in ontologies. More details about views and how to implement them is given in section 6.3. Especially method templates have been applied in order to fill our assertional box (ABox), but also as a structural source for the terminological box (TBox).
Next to declarative knowledge, procedural knowledge from the flowcharts and templates can be stored in the ontologies by using the depicted relations between the methods (*cf.* Figure 4.6). More complex knowledge, like decisions in flow charts, can be embedded by integrating process meta models. These meta models, however, are more important for representing the black box information inside the methods, which are not considered in our method framework. BRs, on the other hand, are mostly declarative, used to formalize the behavior for individual situations and thus do not explicitly represent control flows. The knowledge for describing our method framework and the inner method knowledge, both represented by Semantic technologies, can be interconnected, though. Nevertheless, these are two different use cases and for maintenance reasons, both knowledge repositories should be kept separate. The analyses, introduced in chapter 6, mainly deal with the knowledge represented in the method framework, supplemented by Enterprise Architecture Management (EAM)-specific knowledge, but specific SPARQL queries might also make use of inner method knowledge.
The kind of formulations chosen to represent a business' methods is influenced by sundry factors. Next to the factors introduced in section 4.2.1 (the user's knowledge, experiences, skills, tendencies, ...) and the user's intellectual capacity, it is important to choose a method based on its demanded reliability, the desired degree of freedom when working with the method as well as its related process and intended result (Weigt 2008; Mueller 1990).

4.4. Method Metrics

The so far introduced method model allows observing and comparing methods based on their semantic context information. In this section we want to extend the method model with metrics. Metrics help the user to decide which method to apply in the scenario at hand, for instance, by providing process KPIs like costs, duration, ease-of-use, or quality of a method. The selection procedure based on these criteria is explained in section 6.4.2.
Next to selection criteria, the metrics can act as indicator for strengths and weaknesses in the company's method framework and allows detecting gaps, i.e., processes that are scarcely supported by methods. This information is valuable for the strategic method development. The model can also be extended with arbitrary execution or evolution quality attributes (the so called "ilities"), like usability, reliability, maintainability, manageability etc.
Which kind of non-functional requirements to choose is facultative and up to the company's KEs and method engineers.
Next to the planning and selection aspect, the metrics allow the comparison of alternative methods and allow formalizing statements about the methods' maturities. The maturity can be consulted for either comparing methods among each other or for predicating the company's overall method performance and maturity. Further metrics can be applied to judge KPIs like the process integration grade or the quality of the methods' in- and outputs.

4.4.1. Method Maturity

Maturity models exist for sundry domains with various characteristics and extensions.
Prominent process and business maturity models offer ideas that are also well suited for statements about a method maturity. We introduce aspects of these maturity models and adapt and integrate the relevant parts in our method model in order to make the methods assessable and thus comparable.
Maturity models are used to assess the *modus operandi* of companies in project management, business process management, enterprise architecture management or software and system development processes. For this purpose, the models offer a segmentation into different maturity grades. The maturity grades are then

used to benchmark the company's performance (Jacobs 2014), for instance, to rate how well business processes have been defined and how consistently they are applied.
However, this maturity grade just reflects an abstract method maturity (*cf.* section 4.4.2). When assessing methods, various factors have an influence on the maturity grade, depicted in Table 4.3.

Assessment aspect	**Assessment criteria**
Problem definition	Generality, conditions of practice
Validity	Formal (data), empiric (structure and behavior), pragmatic (purpose)
Effectiveness	Solution probability, solution tractability, *(solution quality)*
Efficiency	Implementation and operation effort, tool support, *(data supply)*
Usability	Documentation, comprehensibility, transferability, *(application result)*

Table 4.3.: Method assessment aspects and criteria. Based on Meißner (2006).

These aspects and criteria have to be taken into account, when rating a method's maturity. Additional "ilities" can be taken into consideration, as well.
We decided not to assess these fine-grained aspects individually but to use an abstract method maturity, because the overhead for entering and maintaining these values would be too high. However, the model can be expanded and customized when more detail becomes necessary. Besides, we further partitioned the maturity grade into input, output and process integration grades.
As a consequence, parts of the efficiency aspect, i.e., the data supply, does not influence our abstract method maturity. This is also true for the solution quality, which is rated as the output quality which also includes the application result criterion (if the result is usable and further applicable).

Comparison with prominent Maturity Models

Besides, prominent maturity models, like Capability Maturity Model Integration (CMMI) and Software Process Improvement and Capability Determination (SPICE), offer predefined standard processes that have to be implemented in order to achieve a higher ranking. For example, both CMMI and Automotive

SPICE, expect the enterprises to implement a project management process (Jacobs 2014). Thereby, CMMI only defines *what* has to be done to achieve a good process and which procedures offer an added value. It does not define *how* the processes and procedures have to be shaped and organized in detail, though.
CMMI provides various views that allow the organizations to follow different goals in order to improve: the *staged* representation that offers five maturity stages for sundry predefined areas of the company and the *continuous* structure which is more flexible because it allows the organization to focus on specific processes.
A higher rating means, that a process has been better implemented taking into account criteria like the quality of documentation and its adherence during the execution. Other criteria concern the processes' dissemination, i.e., if it is only occasionally used or if it has been communicated and applied company-wide. Furthermore, the measurability is an important factor, associated with KPIs for the process management.
We want to adapt existing metrics and evaluation methods in order to support statements about the maturity of methods in the PDPs with our method model. The application of maturity metrics allow method engineers and deciders to detect weaknesses in their overall method architecture. Besides, a comparison of alternative methods, for example a virtual and a physical method, for a specific task based on the maturity indicates the quality of the method's and hence the related process action's outcome.
In contrast to the five stages of the staged approach, i.e., initial, managed, defined, quantitatively managed and optimizing, which represent a general stage of the company, the six capability levels of the continuous approach can be partially reused for the assessment of methods.
The standard SPICE also offers a stage model which helps to grade a company. It originates from the assessment of software development but today it is more widely used as a standard for the assessment of enterprises, especially for the process improvement and capability determination. For the assessment of processes, SPICE adduces ISO norms, such as ISO 12207, which enriches processes with predefined properties:

- Allocation of a name or identification
- Description of the process' input and output
- Description of the process' goal
- Description of the process steps

On the basis of these properties, processes can be checked for consistency, for example, if all required process input resources are properly allocated. It is reasonable to incorporate the four items for the definition of methods as well.
Therefore, our ontology offers concepts to annotate these properties. Furthermore, the existence of these items can be queried by KEs in order to generate metrics about the KB's completeness and consequently, its quality (*cf.* section 6.4.3).
Concepts for the method's goal and steps (procedure) have already been introduced in section 4.2.2. The method's in- and output resources, as well as a detailed method description, are introduced in 4.6. Furthermore, the quality of the diverse method inputs and expected outputs can be assessed with the metrics ontology that is depicted in section 4.4.3.

4.4.2. Concrete vs. Abstract Methods

When talking about a method's maturity, quality or other concerns, we have to take into account, that a method can be applied at several points in a company's processes. Usually, the quality of data or resources rises over time, when the product itself is becoming more and more ready for series production. Assuming that we have a method A in our method KB which can be applied at two different points of time in the PDP, it is conceivable that this method is conducted using input resources of a differing quality. Vice versa, the method's application is resulting in similar, but of differing quality, resources which have to be available at a specific point of time in the process, e.g., a milestone. Nonetheless, it is still the same method, supported by the same tools and performed by similar procedures.
Independently, a method can also be applied with the same, or at least the same quality and kind of, resources multiple times in a process. In order to distinguish between these (concrete) instances of a method and the general (abstract) method they have been derived from, we have introduced the concepts `Concrete Method` and `Abstract Method`. This circumstance is illustrated in Figure 4.8.
Both methods, `Concrete Method` A_1 and `Concrete Method` A_2, are derived from `Abstract Method` A. That means, that A, A_1 and A_2 are the same methods – using the same tools, endeavor to achieve analogical goals and applying similar procedures. The difference between these concrete methods is, that A_1 is applied before *Milestone x*, because its output, `Resource` R_1, is defined to be available from this point of time in the process. `Concrete Method` A_2, on the other hand, has to be

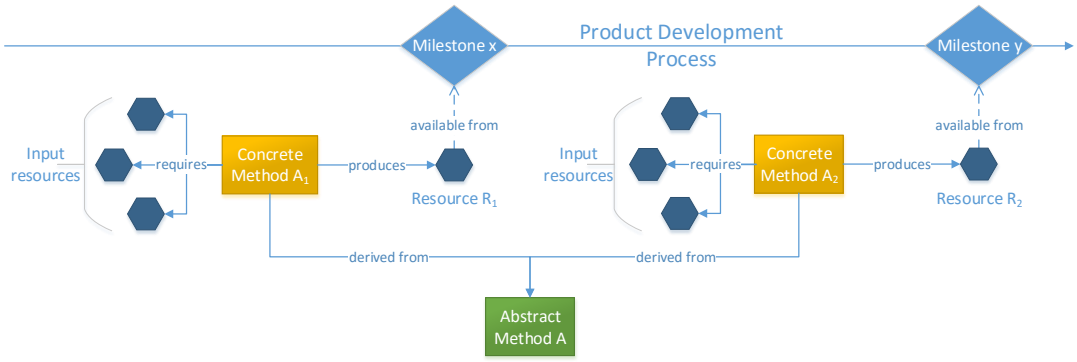


Figure 4.8.: Concrete vs. Abstract Methods.

applied before *Milestone y*, in another task, because of the provision of `Resource` R_2. Their input resources are of the same kind, but in this example it is assumed, that the input resources of `Concrete Method` A_2 are of a better quality, and thus, the output quality of `Concrete Method` A_2 is higher as well, assuming that the remaining boundary conditions are equal. For example, an input parameter for `Concrete Method` A_1 could have been estimated in the beginning of the process. Later on, the method can be repeated with real physically measured values.

A third important quality of a *Concrete Method*, next to its input and output quality, is its degree of implementation or integration in its process action. "Even when methods are applied, the design performance can still be low because of poor use of methods or the quality of the method itself. Low performance can be caused by a mismatch between characteristics of the chosen method and the task or problem at hand, or due to incorrect timing in the process" (Badke-Schaub, Daalhuizen, and Roozenburg 2011).

A high rating in the process quality (*cf.* Figure 4.9) means, that the method is completely and consistently integrated in a process action. This value can either be registered manually into the ontology, i.e., by an expert's experience, or by choosing an appropriate metric defined in an appropriate maturity model (*cf.* section 4.4.1). Besides, its application is well defined and the method's documentation is continuously adhered to. The rating's value is usually based on a DE's assessment. The remaining lower rating stages represent a decreasing quality in consistency, verification and definition culminating in an undefined status.

Applying an abstract method in a process, thus creating a concrete method, is also a decision on what procedures are being used if the method can be executed in alternative ways. This comes along with another important distinction between

an abstract method's concrete instances: they can differ in duration as well as in their caused costs.

4.4.3. Metrics Ontology

The ontologies depicted in the figures 4.9 and 4.10 take into account the distinction between concrete and abstract methods, explained in the previous section 4.4.2. `Concrete Methods` possess a `Concrete Method Maturity` while `Abstract Methods` feature a relation to the class `Abstract Method Maturity`, respectively.

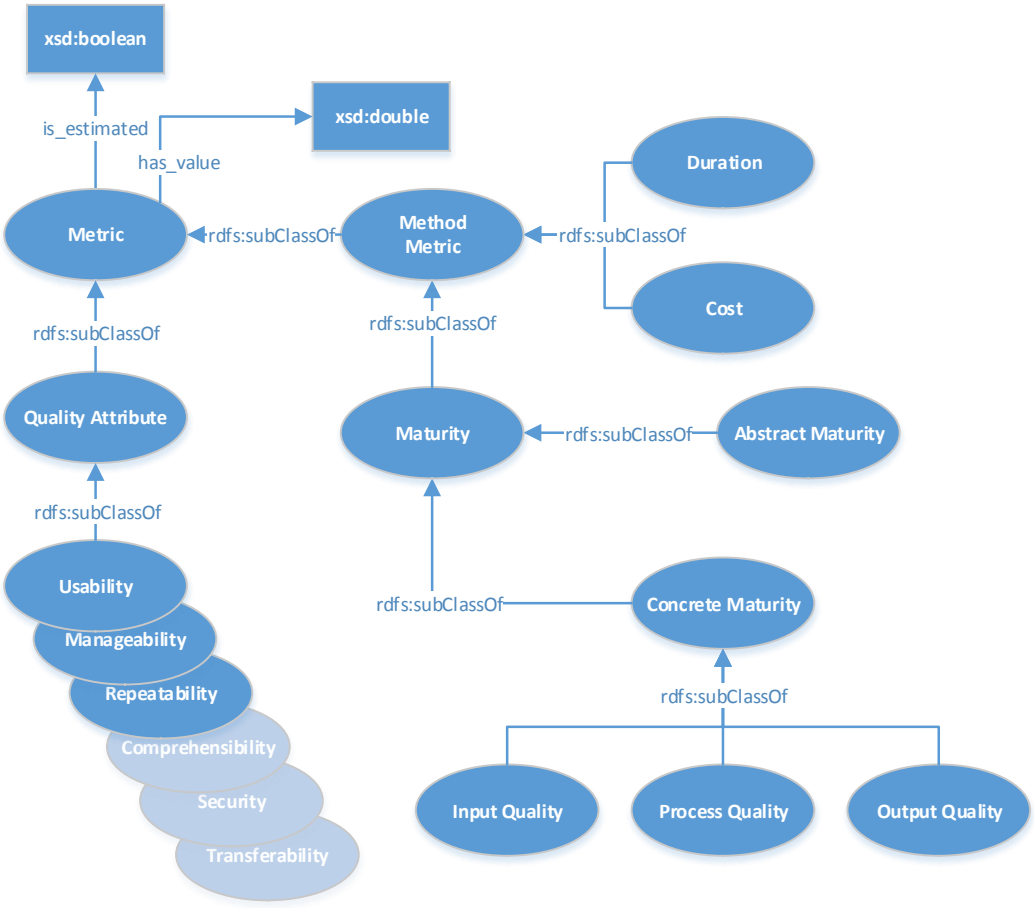


Figure 4.9.: Method Metrics Ontology.

By importing the metrics ontology into our method ontology, three different qualities can be assigned to a concrete method: the input, output and the process integration quality. Both, the `Concrete Method` and the `Abstract Method`, are of

the type `Method`, so they inherit the object properties `has_successor`, `has_predecessor`, `derived_from`, `is_component_of` and `is_alternative_to` which allows defining the relations between all the methods in the KB in a fine-grained way (*cf.* Figure 4.6).

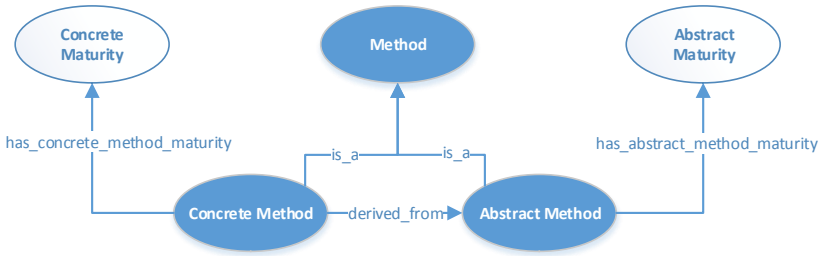


Figure 4.10.: Method ontology extract with maturities.

Next to the three `Concrete Method`'s qualities (`Input Quality`, `Process Quality` and `Output Quality`), the overall `Abstract Maturity` can be assigned to an `Abstract Method`. This maturity expresses the method's quality in general. We decided to use a five stage maturity grade ranging from very high maturity, which means that the method delivers optimal results under repeatable conditions, to very low maturity that stands for an unusable method in the case studies (*cf.* section 7.5). However, this scale can be customized to the company's needs if required. The depicted data property `has_value` with the range *xsd:double* can be generally used for any metrics. Other metric ontologies, for instance for the domain of Business Process Analysis (BPA), make use of more complex concepts, like the *Unit of Measure* which allows assigning units like kilograms or pounds to a value (Pedrinaci and Domingue 2009). For our purpose it is sufficient to assess the quality attributes and maturities with a more simple value. Besides, an extension can be easily realized.

The remaining metrics `Duration` and `Costs` from Figure 4.9 indicate the method's time consumption and the predicted financial expenses. These concepts are explained in more detail in section 6.4 when describing the analyses that can be conducted with our method framework.

When the respective KE fills the ontology it is often the case, that not all information about the metrics is already measured or agreed upon. In this case, the Boolean data property `is_estimated` indicates these entries' reliability.

The ontology extract, depicted in Figure 4.10, shows how to relate the core method ontology's concepts (filled concepts, *cf.* section 4.3) with the method metrics

ontology's concepts (pale background). Likewise, arbitrary quality attributes, e.g., `Usability` or `Manageability`, and method metrics like `Duration` and `Cost` can be related to a `Concrete Method`.

4.5. Linking Enterprise Data

A basic prerequisite for comparing and monitoring methods, that can be performed at a given time in a business process, is the information about their required input resources – whether they exist and are available. In a big enterprise this information is distributed among many IT systems and has to be linked to our method ontology in order to make use of them. This circumstance is exemplified in Figure 4.11, illustrating different layers: enterprise knowledge, e.g., an Enterprise Architecture (EA) model, domain knowledge (*here:* method knowledge), annotations and categorization and finally legacy data and documents at the bottom layer.
Additionally, the method ontologies can be enhanced by linking thesauri, taxonomies or other ontologies which improves the entities' understandability and adds further knowledge. Next to private, company-specific ontologies, companies may also benefit from linking their knowledge with public, community standard ontologies available from the web (Lambrix and Tan 2007), e.g., DBpedia (Lehmann et al. 2014) or Freebase (Bollacker et al. 2008).
Mapping these different sources of knowledge with semantic technology – each entity clearly recognizable via its Internationalized Resource Identifier (IRI) – is known as the concept of "Linked Enterprise Data" (Wood 2010).
Acquiring not yet formalized knowledge like in textual documents or knowledge that is not even written down already has been discussed in section 2.5. This section deals with mapping knowledge that is already available in some formalized way, i.e., in existing domain ontologies (DOs), well-structured data like spreadsheets (Comma Separated Value (CSV), Excel, Extensible Markup Language (XML), ...) or relational databases. The integration of heterogeneous information for this thesis has been evaluated in a case study, matching different Bill of Materials (BOMs) from different CAx disciplines (*cf.* section 7.3) using Frame Logic (F-Logic) to express the mapping rules and by integrating spreadsheet data and relational database tables in the application ontology of the PDD case study (*cf.* section 7.5).

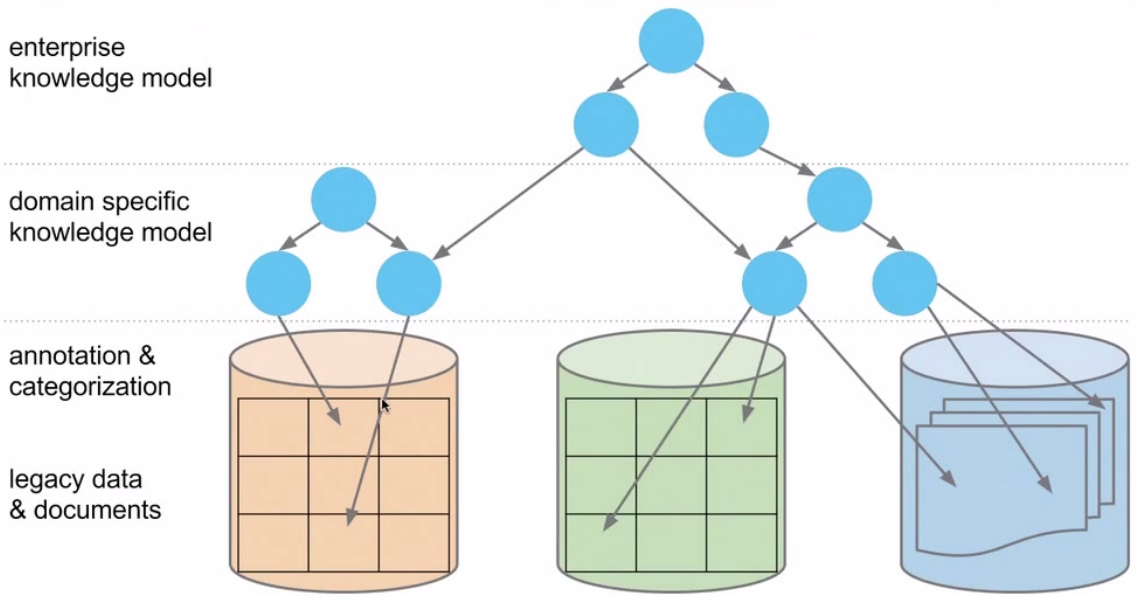


Figure 4.11.: Information architecture example (Blumauer 2014).

4.5.1. Ontology Matching Strategies

Matching heterogeneous resource models with semantic technologies is a promising approach and a current research topic. There are a multitude of reasons why ontology matching is beneficial for a company (Lambrix and Tan 2007). If ontologies in the company are already available it is a good idea to reuse them if the included information is overlapping. This way, applications can cover input from multiple domains and present this input with different view points. Matching different ontologies can be achieved by roughly two different approaches: semantic interlinking via conceptual alignment or via conceptual merging (Zuo 2006). Ontology matching can be used for extending existing ontologies, merge multiple smaller ones or as a basis for creating new ontologies. Furthermore, it might also be a good idea to reuse publicly available ontologies, for instance, the Friend of a Friend (FOAF) ontology (Brickley and L. Miller 2014) for representing person-based information or DBpedia (Lehmann et al. 2014) which is formalized knowledge from the Wikipedia project in Resource Description Framework (RDF).

A number of different matching strategies can be applied in order to create the desired KB for an application, for instance, by utilizing popular and established upper ontologies, like UMBEL (Giasson and Bergman 2015). The matching strategy means, that relations between two or more ontologies are defined, resulting

in an *alignment*. More specifically, schema matching is defined as "establishing *correspondences* between elements of the source and target schemas" (Chomicki 2008), while Chomicki defines *schema mapping* as the "generation of assertions (queries) from schema correspondences" (ibid.).

Definition 4.10 (*Ontology Mapping*)

"Given two ontologies O_1 and O_2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a corresponding entity, which has the same intended meaning, in ontology O_2" (Ehrig and Staab 2004).

Merging, on the other hand, means to create a new ontology by combining multiple source ontologies.
Matching can either be achieved manually or (semi-)automatically, for example, by comparing structures, by using linguistic information for the matching, by applying instance-based, constraint-based, matching based on auxiliary information or a combination of these strategies. The schema-based matching strategies can be classified into three different kind of categories (*cf.* Figure 4.12):

Terminological (Strings): The ontology descriptions usually comprise strings in the entities' labels, comments, attributes etc.

Structural (Structure): Structural information can be matched by analyzing the entities' types and relations with other entities.

Semantic (Models): Correspondences can be deduced by semantic interpretation, e.g., the use of logical reasoning. This procedure results in models.

Next to the schema-based matching strategies, matching via extensional knowledge, i.e., by data instances which "constitute the actual population of an ontology", is a fourth option for matching different ontologies (Shvaiko and Euzenat 2013). Technologically, the matching can be achieved by using classical applications, for instance, written in Java or some arbitrary programming language, or by applying semantic technologies like ontologies or rules, e.g., OWL, SPARQL, F-Logic or ObjectLogic, or a combination thereof.
Shvaiko and Euzenat have compared the state-of-the-art of ontology matching in a recent survey: "These results show a measurable improvement in the field, the speed of which is albeit slowing down" (ibid.). They state, that overcoming a semantic heterogeneity can be achieved with two consecutive methods (ibid.):

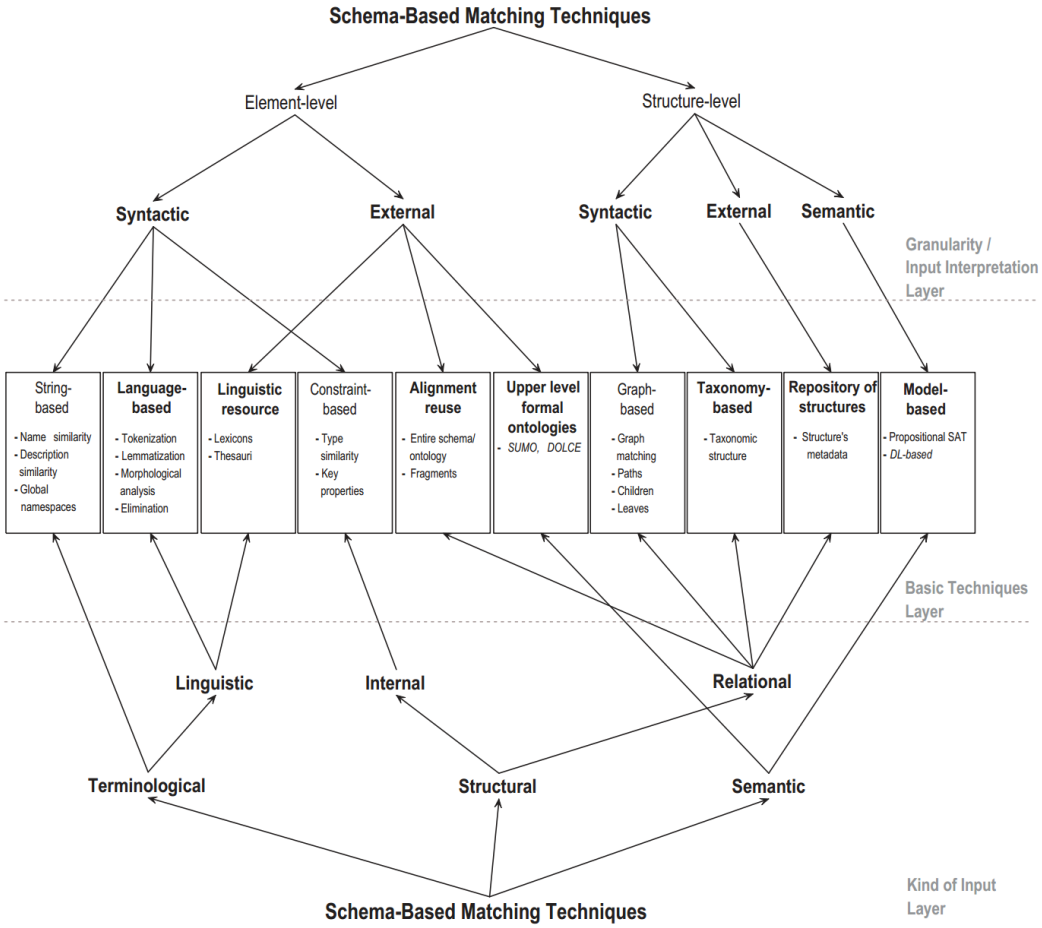


Figure 4.12.: "A retained classification of elementary schema-based matching approaches" (Shvaiko and Euzenat 2005).

i.) matching entities to determine an alignment, i.e., a set of correspondences, and

ii.) interpreting an alignment according to application needs, such as data translation or query answering.

Their survey focuses on the first of the two steps. In this thesis however, we concentrate more on the second step, i.e., query answering and data representation in applications.

Our primary goal in the first step is to find matching entities in two different ontologies, i.e., a *correspondence* $\langle id, e_1, e_2, r \rangle$ which is defined as a quadruple as explained in Definition 4.11 (ibid.).

Definition 4.11 (*Correspondence*)

A correspondence is defined as a quadruple: $\langle id, e_1, e_2, r \rangle$, such that:

- *the id represents an identifier for the given correspondence;*
- *e_1 and e_2 represent entities of the two different ontologies, for instance, classes and properties, respectively;*
- *and r represents a given relation, for example, an equivalence ($=$), a disjointness ($\perp$) or a more general ($\sqsupseteq$) relation, holding between the two entities e_1 and e_2.*

The set of correspondences between all matching entities of the two ontologies is the *alignment*. "Alignments can be of various cardinalities: *1:1* (one-to-one), *1:m* (one-to-many), *n:1* (many-to-one) or *n:m* (many-to-many)" (Shvaiko and Euzenat 2013).
It is often a good idea to express the confidence, i.e., the likelihood of the relations, in the ontology as a metadata element as seen in Figure 4.13.

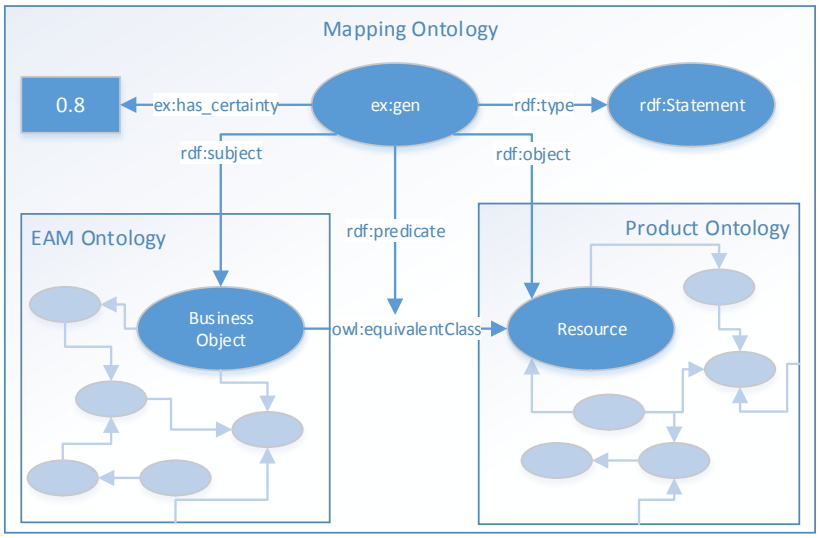


Figure 4.13.: Reification example, expressing the certainty about an RDF statement.

This way, co-workers, applications and other KEs can estimate the correctness of the matchings made. When using an RDF ontology, this confidence can be applied to an RDF statement using reification. Next to reification, there are many other methods how to accomplish meta statements with RDF and OWL, like the use of Singleton Properties (Nguyen, Bodenreider, and Sheth 2014), using named

graphs, blank nodes or by adding classes about relationships. Other meta information, like the author of the statement, the date of creation etc. can be added with any of these procedures. However, the best suitable method for creating such statements about statements always depends on the application.
Usually, the matching for the data integration is done during the *design time*, like the concepts illustrated above. Nevertheless, Shvaiko and Euzenat (2013) also introduce concepts for the data integration of ontologies during the *run time* which is useful when the underlying models are unknown or can often change.

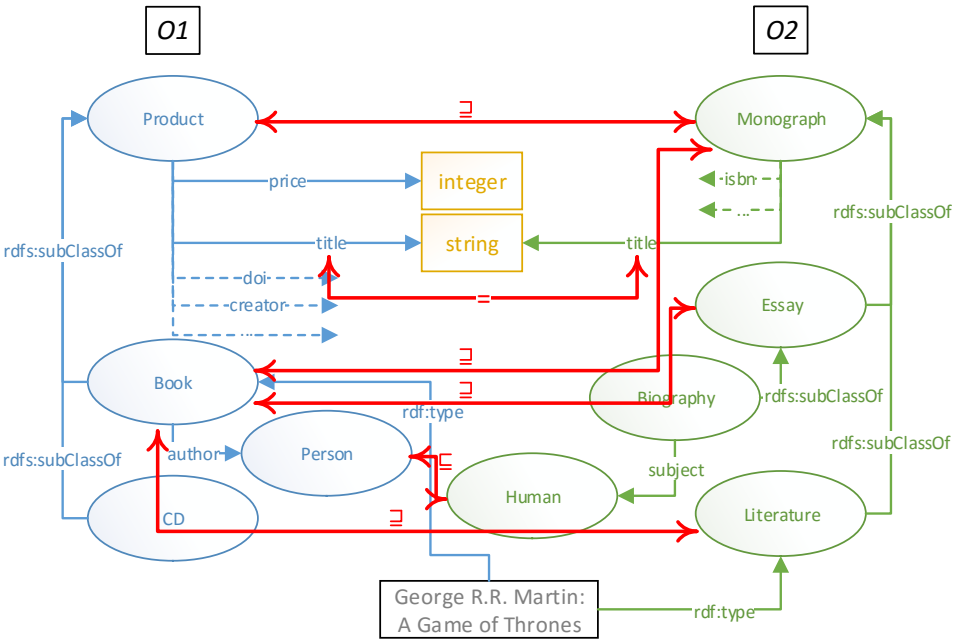


Figure 4.14.: An exemplary alignment (in red) between two simple ontologies. Based on Shvaiko and Euzenat (2013).

The two ontologies O_1 (in blue) and O_2 (in green) and the overlayed matching depicted in Figure 4.14 illustrate an alignment. The thick red arrows between entities of the two ontologies represent a correspondence. This correspondence is annotated with the type of relation (either a concept *inclusion* ($\sqsupseteq$) or an *equivalence* relation ($=$)).
However, before we can find and define correspondences between semantic heterogeneous ontology entities, we often need to transform these entities so that they resemble the same format. This is called a *complex match*. For example, the two ontologies describing BOMs that include a list of parts from the evaluated case study (*cf.* section 7.3), cannot be directly matched as seen in Figure 4.15.

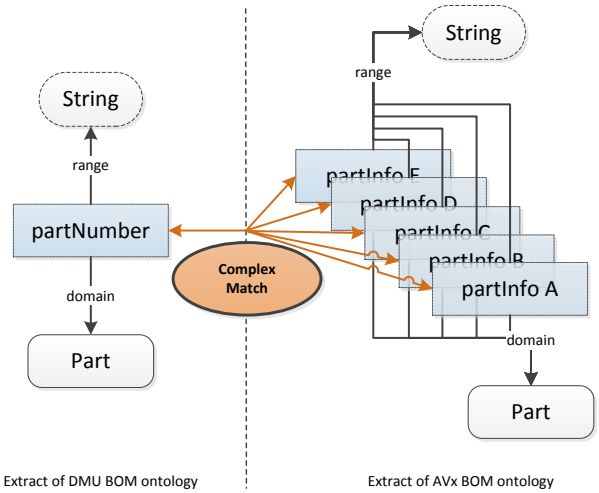


Figure 4.15.: Complex Match between Digital MockUp (DMU) and AVx Ontology.

The part number in the AVx BOM is present as a variety of different partial data properties. The DMU BOM consists of a single data property that represents the whole part number. Before we are able to match these part numbers with an equivalence (=) we need to find a way to express part numbers in the AVx BOM using the same format as in the DMU BOM. Similar difficulties arise when matching ontologies that contain different units, e.g., metric vs. imperial system, or use the same term for describing different semantics, for instance a property *area* can represent a surface area or a location. Therefore, the very first operational step is to align the various data entities so that the source representation is translated into a target representation. The target representation should be mutually agreed on by the different stakeholders of the proposed application and synchronized to the previously identified data representation of the application itself.
In the next section we will show how to use rules in order to transform ontological knowledge into the desired target format.

4.5.2. Using Rules for Ontology Alignment

With SPARQL 1.1 (Harris and Seaborne 2013), the current World Wide Web Consortium (W3C) recommendation, it is possible to use *CONSTRUCT* to generate new graphs that consist of transformed representations, which have been pre-

viously unmatchable, and carried over data of prior compatible representations. The two examples in 4.1 and 4.2 show how to make simple alterations by renaming properties or classes. The source ontology uses the namespace variable "*avx*", whereas the newly generated graph uses a namespace abbreviated by "*cax*". In the first statement, the new relation *cax:new_property* between two entities (subject ?*s* and object ?*o*) is created, if the relation *avx:old_property* already exists between those two entities. The second example states, that all subjects ?*s* that are a member of the class *avx:AVxPart* are now a member of the class *cax:Part*.

```
CONSTRUCT {
  ?s cax:new_property ?o
} WHERE {
  ?s avx:old_property ?o
}
```

Listing 4.1: Rename property with SPARQL *CONSTRUCT*.

```
CONSTRUCT {
  ?s a cax:Part
} WHERE {
  ?s a avx:AVxPart
}
```

Listing 4.2: Rename class with SPARQL *CONSTRUCT*.

The *WHERE* clause in the SPARQL statement can describe much more complex conditions for the matching entities as shown in the two examples above, like using filters or simple mathematical operations. The example in Listing 4.3 shows a *CONSTRUCT*, that applies a mathematical operation with the *BIND* form and then filters the calculated values with the *FILTER*. In the example, a runtime given in hour format is converted into a runtime expressed in days, if the runtime is greater than 72 hours.

```
CONSTRUCT {
  ?s cax:runtimeInDays ?timeInDays .
} WHERE {
  ?s avx:runtime ?timeInHours .
  BIND(CEIL(?timeInHours / 24) As ?timeInDays) .
  FILTER (?timeInHours > 72)
}
```

Listing 4.3: Transform and filter values with SPARQL *CONSTRUCT*.

SPARQL 1.1 offers a wide range of functions that can be used to transform or filter values, e.g., functions on RDF Terms (*isURI*, *isLiteral*, ...), String operations (*STRLEN*, *SUBSTR*, *CONCAT*, regular expressions, ...), functions on Numerics (*abs*, *ceil*, ...), Dates and Times (*now*, *year*, ...), Hash Functions (*MD5*, *SHA1*, ...) or XPath Constructor Functions (*xsd:double*, ...). Offering this diversified amount of operations, a lot of matching scenarios can be processed with SPARQL alone. Secondly, it is part of the Semantic Web (SW) Stack and thus should be used as the recommended query language. However, we will encounter scenarios in the second case study (*cf.* section 7.5) where more complex statements, i.e., using recursion and looping or branching constructs, would be necessary.
Alternatives for transforming the knowledge using rules are, for instance, Jena Rules (The Apache Software Foundation 2014), Semantic Web Rule Language (SWRL), F-Logic or ObjectLogic. The Listing A.3 on page 319, done during the BOM evaluation, depicts how to use F-Logic for finding the correspondences between the part number in the AVx und DMU BOM depicted in the previously shown Figure 4.15.

4.5.3. Integrating Databases

In a heterogeneous enterprise IT landscape, a big part of the knowledge about methods and their context is usually stored in relational databases. These databases are connected to a variety of application and are involved in a multitude of workflows. This way, the data is kept up-to-date, however, the semantics of the data seldom is part of the database but defined in the application that integrates it. Because we want to use up-to-date data and do not intend to modify existing data within our method ontology application, we need a way to integrate this information into our ontology. Mirroring the data, i.e., building a huge ABox by

performing a massive dump, is not an option, since maintaining the instances would be an endless effort. Thus, we need an interface that connects our model to the databases and queries its contents in real-time, "i.e., only the data needed to answer the user's query are retrieved from the sources" (Ghawi and Cullot 2007). The downside of this approach is a lesser performance, because we have a delay in query processing (Rodríguez, Rodriguez, and Gómez-Pérez 2006), but dealing with fresh information is worth it.
A classification of the different database-to-ontology mapping scenarios is depicted in Figure 4.16. For integrating the necessary data into our method ontology, we pursue both mapping approaches: we create new ontologies from existing databases and then want to use these newly generated mapping ontologies to augment our existing method ontology.

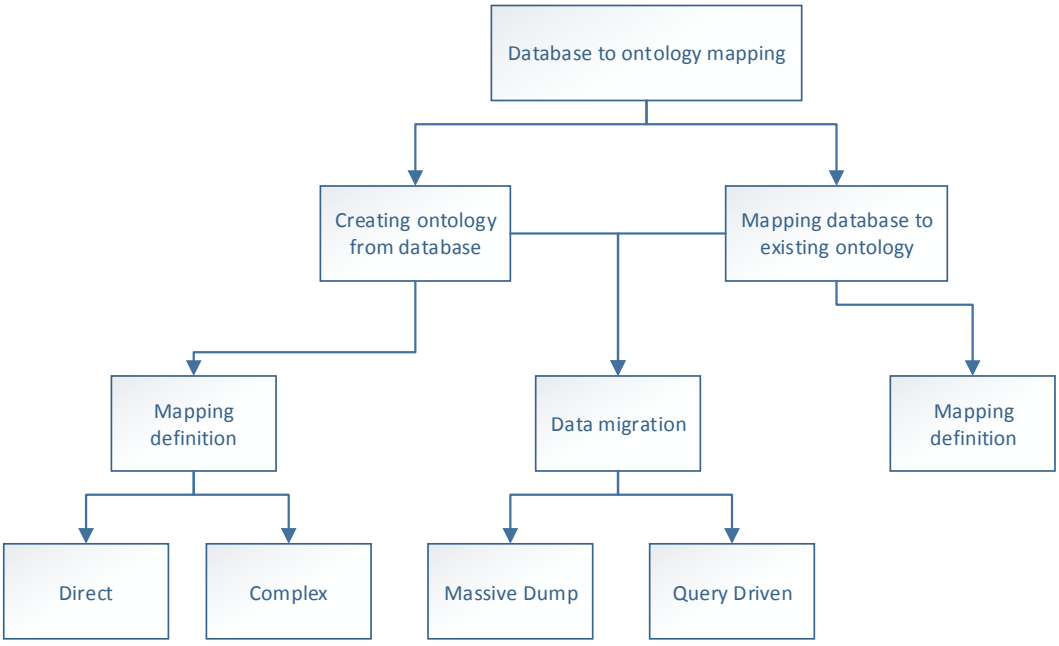


Figure 4.16.: Classification of database-to-ontology mapping approaches (Ghawi and Cullot 2007).

Before we can use an adapter to query the databases, however, we need to understand the meaning of the tables, columns and their comprised data. An effective approach is to create a mapping ontology, i.e., the TBox, that reflects the schema of a database. Database tables are described by ontology classes and the database columns by properties. Mapping columns is often a straightforward approach. The type of the columns in the database table are transformed to an XML Schema

Definition (XSD) type (*String*, *Integer*, *Double*, ...) using a data property. Columns that represent identifiers to link between different database tables and the corresponding constraints can be expressed by introducing new object properties that connect the appropriate classes.

Creating an ontology model that corresponds to a given database schema by introducing 1-to-1 correspondences can be achieved automatically, e.g., with DB2-OWL (Ghawi and Cullot 2007). However, the semantics that are not included in the database have to be collected using other methods. Besides, a further refinement of the ontology is suggested. This is mostly a manual effort and includes interviewing experts or analyzing documentations and the applications. Furthermore, structural changes in the source database have to be traced by either introducing new maintenance processes in the enterprise or making use of already established ones, e.g., via an EAM.

This database mapping ontology is then matched to our original method ontology with techniques described in the previous sections.

Fortunately, using SWTs in order to adapt existing SQL databases is a mature concept, today, and many software solutions, that perform this task, exist. That way, the already established systems can be kept running while making use of the data in a modern ontology-based application. An exemplary diagram of a connection between an SQL database and SWTs is depicted in Figure 4.17 which was presented in a talk of Berners-Lee.

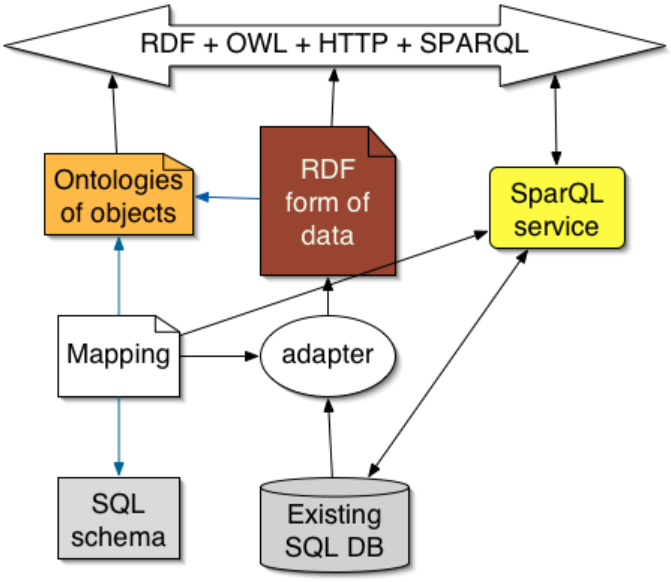


Figure 4.17.: Adapting SQL databases with SWT (Berners-Lee 2006).

4.6. Integrating the Method Ontologies

In this section, we will combine our previously introduced method ontologies, i.e., the core method ontology (section 4.3) and the metrics ontology (section 4.4.3), with the remaining essential ontologies in order to model and analyze a company's method landscape as a whole. Later on, in chapter 6, we will also incorporate an EAM ontology, along with suitable extensions for our purpose.
The requirements for our method model have already been stated, in addition to possible matching techniques. Therefore, one ontology, that we will model and include, is a resource ontology, covering the required inputs and produced outputs of our methods. It will be illustrated in section 4.6.1.
Next to the method' inputs and outputs, a method's usage and application scenarios have to be described. Hence, we illustrate how to add an explanation to a method, for instance, by attaching documents (see section 4.6.2).
The final important subsection will deal with the usage of ontologies as a Controlled Vocabulary (CV), shown in section 4.6.3. This way, multiple labels can be annotated to all the ontology's entities which makes them generally intelligible, e.g., by enabling a multilingual use or making the semantics more comprehensible for people with differing professional backgrounds.
One of the big advantages of using SWTs is having this vast amount of publicly available community ontologies that can be used as extensions, geared to the company's needs. We can only cover a small sample of these KBs, like FOAF (Brickley and L. Miller 2014), Bibliographic Ontology Specification (bibo) (D'Arcus and Giasson 2009), Simple Knowledge Organization System (SKOS) (Isaac and Summers 2009), UMBEL (Giasson and Bergman 2015), schema.org[4] or Dublin Core (DC) (Dublin Core Metadata Initiative 2014) for the already mentioned product, CV and description ontologies. Nevertheless, other convenient and well-known ontologies, like *vs*[5], *vann*[6](Davis 2005) or ontologies about spatial or temporal knowledge, should be added as required. This knowledge from the above paragraphs is outlined in the method ontology's architecture in Figure 4.18. The majority of the concepts and relations, for instance, the resources and maturities, have already been discussed. Our core method ontology is integrated as an upper ontology in this architecture. The remaining ontologies, e.g., a process, re-

[4] http://schema.org/; last visited on 08/11/2015.
[5] an ontology to add a status to a vocabulary term (*stable*, *unstable*, ...) – http://www.w3.org/2003/06/sw-vocab-status/ns (retrieved 08/04/2014)
[6] Allows the annotation of vocabularies with descriptions, examples and usage notes.

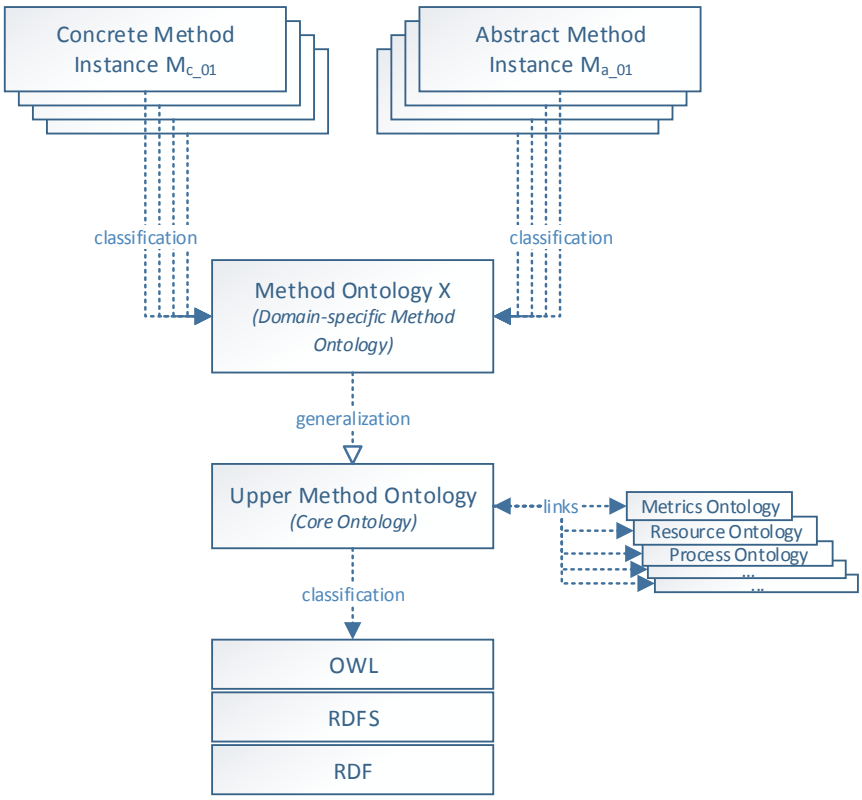


Figure 4.18.: The method ontology's architecture.

source and metrics ontology, are mapped to the core ontology, for example with *owl:imports*. Another possibility for mapping the different ontologies is the usage of feature models (Langermeier, Rosina, et al. 2013; Langermeier, Driessen, et al. 2014). These feature models contain the information about the possible, required and prohibited mappings between the single ontologies. Depending on the domains, a KE wants to cover in his KB, various conceivable combinations are feasible, particularly if different ontology versions or even competing ontologies are at our disposal. For example, spatial knowledge can either be included with the Geopolitical ontology (Food and Agriculture Organization of the United Nations 2014) or GeoNames Ontology (Vatant and Wick 2012).

Independent from the chosen mapping paradigm between our upper ontologies, a company wants to model its own KB, covering their use cases. This DO is depicted in Figure 4.18 as *Method Ontology X*. The DO specializes the concepts of the upper ontologies with domain specific ones and provides the TBox for our specialized ABox, i.e., *Concrete Method Instances* M_{c_u} and *Abstract Method Instances* M_{a_v}.

For instance, it can be used to provide a necessary taxonomy for the domain of CAx methods by introducing new concepts, e.g., `CAE Method`, `CAD Method` and `CAT Method` or a more general method, like *dc:MethodOfInstruction* from DC. Along with new kinds of methods, apposite, more specialized metrics or other arbitrary parameters can be introduced in the same way.
The knowledge for creating an own domain-specific ontology can either be acquired from existing company sources (see section 2.5), like documents, or from experience, by interviewing experts. Another source of information might be an available Body of Knowledge (BoK) which provides taxonomies, definitions and a vocabulary for a specific domain. For example, Software Engineering Body of Knowledge (SWEBOK) (Bourque and Fairley 2014) for a software engineering ontology or Enterprise Architecture Body of Knowledge (EABOK®) (EABOK Consortium 2014) for enterprise architecture knowledge, depending on the company's kind of business domain.

4.6.1. Resources

In accordance with the distinction between concrete and abstract methods (section 4.4.2), we also distinguish between `Concrete Resources` and `Abstract Resources`.

Definition 4.12 (*Abstract Resource*)
An abstract resource R_a is a concept that defines a class of process-independent resources.

Definition 4.13 (*Concrete Resource*)
A concrete resource R_c is a concept that defines a class of resources, applied in a process.

When modeling an `Abstract Method`, the KE can define the type of resources that are required or produced by this method, for instance, a general placeholder parameter for the weight with the unit kilograms or an abstract CAD model name. These types of resources can be classified as an `Abstract Resource` (*cf.* Figure 4.19).
This approach is similar to the modeling of problem solving methods by Motta.

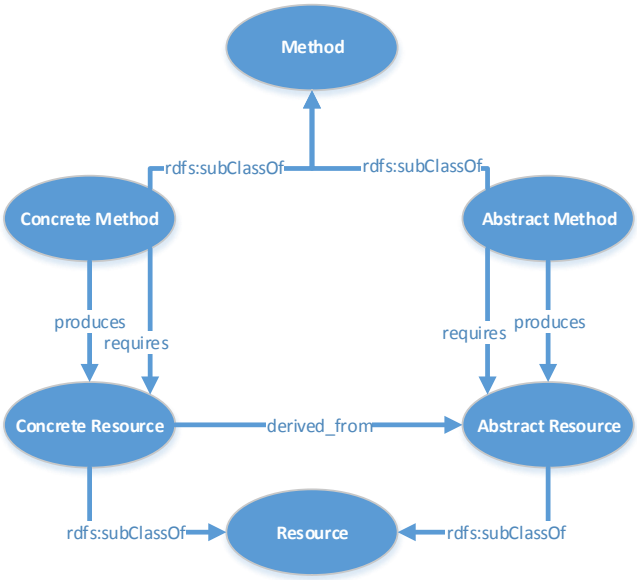
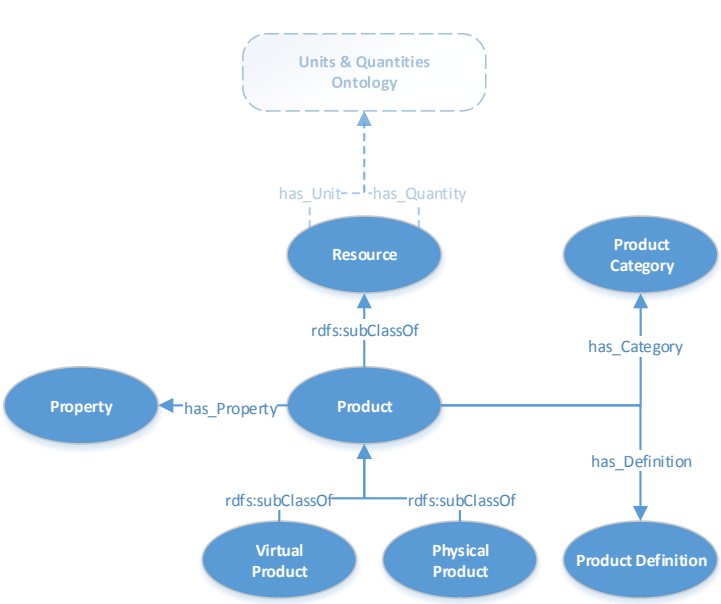
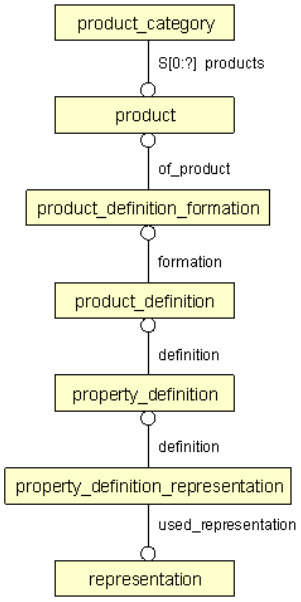


Figure 4.19.: Concrete and Abstract Resources.

He defined the components that are necessary to describe a problem solving method. These are, among others, input and output roles, whereas the input role is "a specification of the various types of knowledge required as an input by a method" (Motta 1999). Vice versa, an output role is defined as "a specification of the types of output produced by a method" (ibid.). The resources in our method ontology comprise, in addition to knowledge, every possible entity that is either necessary to execute a method or the method's produced output. This can be abstract people, i.e., role descriptions, item classes, abstract data descriptions, etc.
But not until the `Concrete Method` instance is modeled and assigned to a process action, the `Concrete Resources` can be named, for example, a specific vehicle part's weight or a particular drawing.
The extract depicted in Figure 4.19 depicts the detailed relationships between `Resources` and `Methods` and thereby refines our previous method model, introduced in section 4.3.
Following current best practices, like the example of W3C Product Modelling Incubator Group (W3PM) (W3PM 2009) and Westphal, Meerkamm, et al. (2009), our resource model, depicted in Figure 4.20a, is influenced by a standard product model, namely STandard for the Exchange of Product model data (STEP) (ISO 2014, AP 214).



(a) Resource ontology.



(b) STEP backbone (W3PM 2009).

Figure 4.20.: Product Models.

Definition 4.14 (*Product*)

Thing or substance produced by a natural or artificial process (ISO 10303-41).

STEP is an International Organization for Standardization (ISO) standard for the description and representation of geometry and product structure and is applied world-wide in various industries, most notably in automotive and aeronautic enterprises. It is used to exchange, share and archive data in sundry applications and systems, like CAD, CAE, DMU and Product Data Management (PDM) systems. This way, we have a product structure that is compatible to many standardized CAx systems which simplifies data exchange and allows us to map our structure to already in place PDM systems.

Like W3PM, we do not model the STEP product structure one-to-one as seen in the juxtaposition in Figure 4.20: primarily, because it is too complex for our purpose but also because it is not semantically well defined and thus not easily modeled in an ontology. For instance, the class `product_definition_formation` in the STEP backbone (*cf.* Figure 4.20b) is described and defined very vaguely in the standard (W3PM 2009).

Additionally, STEP defines a product either as a single physical object or as a class of physical objects, represented by a producer version. In contrast, we also con-

sider `Virtual Products`, which is reflected in our resource ontology (*cf.* Figure 4.20a). Besides, our ontological concepts always represent classes of individuals by definition.

The remaining STEP backbone concepts, the `Property`, the concept `Product` itself, the `Product Category` and the `Product Definition` can be rediscovered in our resource ontology (*cf.* Figure 4.20). A product `Property` represents an inner or outer product's attribute, something that can be tangible, experiencable or required by the customer. Ehrlenspiel and Meerkamm (2013) define a product property as something than can be determined by observation, measurements, common knowledge etc., closely related to a product characteristic (DIN 2330). In the automotive domain, a product property is, for instance, the vehicle's safety, sportiness or off-road capability.

Solely the concept "representation" is not covered in our ontology, because its STEP definition lacks convincing semantics.

Furthermore, in order to be more general and not only cover products, we superinduced the concept `Resource`. Every `Product` is a `Resource`, which allows us to link our methods from the method ontology via the previously introduced concepts `Abstract Resource` and `Concrete Resource` in Figure 4.19.

Like all the introduced ontologies in this thesis, the resource ontology is regarded as an upper ontology that should be extended with a domain specific product structure according to the method ontology's architecture depicted in Figure 4.18 on page 138.

We decided not to model a product's inner structure, like PDM systems do, but to confine ourselves with more abstract concepts, because our method ontology's purpose is still to model only the meta level of the method landscape, i.e., analogous to the method concept, we treat the products as a black box. In other words, we only consider resources required and produced by methods as a whole. These omitted statements are, for example, "a has b as a part" or "a is connected to c". The same applies to cardinalities to express that a "each member of X has *exactly* 2 members of Y as a part" (*cf.* W3PM 2009).

As a concession to complexity and in order to analyze the method's in- and output in a finer way, we indicate the resources' relation to a Unit & Quantities ontology, analogous to the work of W3PM. This way, simple scalar properties can be attached to a resource as seen in Listing 4.4. But also other realizations, like the work of Pedrinaci and Domingue, can be mapped to our ontology (Pedrinaci and Domingue 2009).

```
<Thing rdf:about="#MyVehicle">
 <totalLength>
  <Length><metre>12</metre></Length>
 </totalLength>
</Thing>
```

Listing 4.4: Expressing scales and units in a product ontology. Based on W3PM (2009).

4.6.2. Method Description

A method typically consists of the five bullet points listed below (Lindemann 2009):

Purpose / Goal: A task in the PDP which is supported by the method.
Situation: The scope, problem descriptions and boundary conditions the method is usually appropriate for.
Effect: Effects and side effects, that are attained by executing the method, i.e., the method's output.
Procedure: The performed steps when executing the method.
Tools: Form sheets, check lists, software, test beds, etc.

We have already taken care of the purpose/goal, situation, effect, procedure and tools a method is associated with. Every bullet point is covered by at least one concept in our ontologies. For example, we can deduce where and when a method should be applied in a business process by analyzing its relations to process actions.
Nevertheless, there are information that cannot be covered by our model, or at least, it is not designated for this purpose at the moment. For one thing, the method's detailed procedure is only vaguely embodied. For another thing, information about its boundary conditions, a general problem description etc. is currently not envisaged.
We have decided to not formally model the method's detailed instructions in our meta method ontology for obvious reason: our goal is to capture the meta knowledge about a method's application along with statements about, for instance, its efficiency, inputs, outputs and the method's relations to other domains, such as

processes. Nevertheless, there is doubtlessly a possibility to attach a detailed method description or any other further information.
For a start, this information can be added to each method instance by using annotation properties like *rdfs:comment* or by making use of more expressive metadata vocabularies, like the prominent DC (Dublin Core Metadata Initiative 2014), or by taking advantage of a CV (see section 4.6.3).
Furthermore, it can be partially deduced by analyzing the methods semantic context in the ontology, for instance, with selected SPARQL queries.
Finally, ontologies in conjunction with rules expressing methods and their behavior, i.e., detailed descriptions and instructions, have been introduced in section 2.5. These models can of course be attached to our meta method framework, but should be kept separate, for they have different lifecycles, responsibilities and use cases.
In addition to the options listed above, we introduce the new concept `Document` for good measure as an entity that can hold this information.

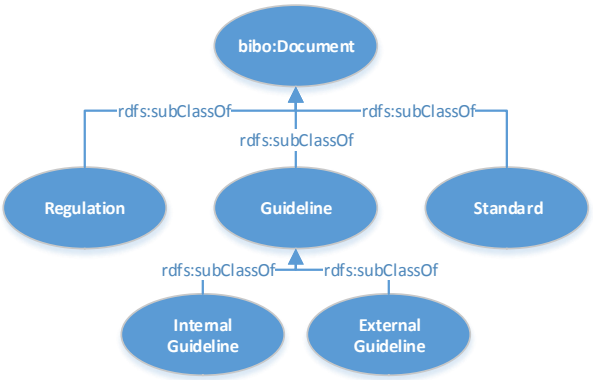


Figure 4.21.: A selection of document types for the method ontology.

This way, the description are reusable and we have the advantage to introduce a possible classification for documents as exemplary depicted with the concepts `Regulation`, `Guideline` and `Standard` in Figure 4.21. Like before, this ontology should be further customized with suitable concepts. The classification helps, because we have use cases where users solely work with one type of document for a method, for example, regulations, law texts, corporate or external policy guidelines or other kind of rules.
It depends on the application, that uses the ontology, how the description itself is deposited. This can either be achieved by the options mentioned above or by pro-

viding references to arbitrary documents in a file system, database or any other repository.
Likewise, `Procedures` can be described by a `Document` as well.
Methods can be described in varying levels of detail. On the one hand, they can be described by a formal method model (Motta 1999). On the other hand, a coarser grained explanation, either textual or illustrated, can be attached via this concept.
The base class `Document` is equal to `bibo:Document` (D'Arcus and Giasson 2009) which again is equivalent to `foaf:Document` by definition: `<owl:equivalentClass rdf:resource="&foaf;Document"/>`. This document can either be electronic or physical (Brickley and L. Miller 2014).
By referencing a regulation or guideline to a geographical ontology's entity, e.g., from the Geopolitical ontology (Food and Agriculture Organization of the United Nations 2014) or GeoNames Ontology (Vatant and Wick 2012), it can be associated with a region or country to include the information where this document has to be considered. For example, when producing goods that are to be shipped world-wide, different methods have to applied in order to guarantee safety or other regulations that are in effect at its place of destination. This association can best be expressed, reusing the DC term `dc:coverage`[7] which deals with "the spatial applicability of the resource, or the jurisdiction under which the resource is relevant" (Dublin Core Metadata Initiative 2014).

4.6.3. Using the Method Ontology as a Controlled Vocabulary

In a big enterprise with its diverse departments, many people with a varying knowledge background have to interact with each other. This circumstance gets even more complicated, when suppliers or other companies want to share and discuss knowledge. People often refer to the same concepts using different terms which is a hindrance in their communication.
An ontology that models the knowledge does not only explain the concepts' coherences and differences semantically, but can act as a CV, too. This means that a different syntactical representation, i.e., terms, can be annotated to each present concept, relation or instance which simplifies managing the heterogeneity of the vocabulary. Even multilingual annotations can be deposited, because RDFS and therefore OWL supports labeling inherently (W3C OWL Working Group 2012).

[7] http://purl.org/dc/terms/coverage

The `rdfs:label` property is primarily used for making an IRI human-readable, but different labels representing a synonymous vocabulary for the very same concept can be annotated as well as seen in Listing 4.5.

```
<owl:Class rdf:about="http://www.example.com/2014/04/crash_test#
    ↪ Sled_Test">
  <rdfs:label xml:lang="de">Schlittenversuch</rdfs:label>
  <rdfs:label xml:lang="en">sled test</rdfs:label>
  <rdfs:label xml:lang="en">sledge test</rdfs:label>
</owl:Class>
```

Listing 4.5: Multiple *rdfs:label* annotations.

The difficulty for the user is to decide which representation to use if multiple ones are available, like the two English representations in the example.
Furthermore, the `owl:deprecated` annotation property with its value set to *"true"^^ xsd:boolean* can be used. It marks an IRI as deprecated, signaling the agent that it should no longer be used, because it is planned to be phased out.
Besides, annotation properties like `rdfs:comment`, `rdfs:seeAlso` and `rdfs:isDefinedBy` can be attached to provide additional information (W3C OWL Working Group 2012).

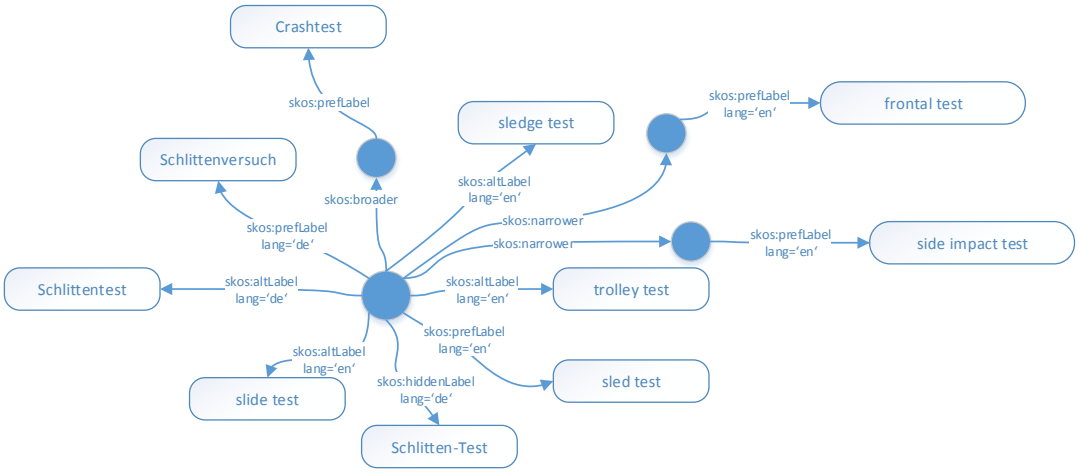


Figure 4.22.: SKOS example.

Because OWL itself only supports the aforementioned basic annotation properties with its shortcomings, we have decided to extend our ontology with SKOS (Isaac and Summers 2009) which is based on RDF. The SKOS model is used to express all kinds of CVs, like taxonomies, thesauri, folksonomies, classification

schemes and other types of concept schemes. It allows labeling concept IRIs with multiple types of labels as seen in the examples Listing 4.6 and Figure 4.22.
The advantage compared to `rdfs:label` is the ability to annotate preferred labels, which expresses the paramount example of the concept's syntactical representation. Besides, as many as necessary alternative labels and even hidden labels that represent misspellings can be annotated. They are used to encoding linguistic variants or unify different vocabularies. This way, the user can search for the concept even with non-standard or deprecated labels while the preferred label comes to the agent's notice. The circumstance is exemplified in the above Listing: the preferred label for the concept `SledTest` is "sled test". Alternative labels are "trolley test" and "sledge test". An annotated hidden label also supports the usage of "slide test" for the concept although this term is not officially allowed in this CV. Of course, SKOS also supports multilingual labeling as seen in the example Listing 4.6. For each language, the `skos:prefLabel`, `skos:altLabel`, etc. can be attached.

```
<rdf:Description ↙
    ↳ rdf:about="http://www.example.com/2014/04/crash_test# ↙
    ↳ SledTest">
  <skos:prefLabel df:lang="en">sled test</skos:prefLabel>
  <skos:prefLabel df:lang="de">Schlittenversuch</skos:prefLabel>
  <skos:altLabel df:lang="en">trolley test</skos:altLabel>
  <skos:altLabel df:lang="en">sledge test</skos:altLabel>
  <skos:altLabel df:lang="de">Schlittentest</skos:altLabel>
  <skos:hiddenLabel df:lang="de">Schlitten-Test</skos:hiddenLabel>
  <skos:hiddenLabel df:lang="en">slide test</skos:hiddenLabel>
  <rdf:type ↙
      ↳ rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
</rdf:Description>
```

Listing 4.6: Using SKOS to organize knowledge. Based on Omrane, Nazarenko, Rosina, et al. (2011).

Linking a SKOS concept with an OWL concept can be achieved through different procedures. First, SKOS, which is defined as OWL Full, i.e., it is undecidable, allows defining a concept not only as a SKOS concept, but also as an OWL concept at the same time like depicted in Listing 4.7.

```
<Concept> rdf:type skos:Concept , owl:Class .
```

Listing 4.7: Concept as *skos:Concept* and *owl:Class*.

This kind of modeling is consistent with the SKOS data model. Since `skos:Concept` is an instance of `owl:Class` as defined in the SKOS data model, this statement is expressed in OWL Full.
If the user wants to strictly use OWL DL, the CV and the OWL DL ontology have to be kept strictly separate. A solution for linking a SKOS concept scheme with an OWL DL ontology is to use annotation properties (Isaac and Summers 2009) which is exemplified in Listing 4.8.

```
ex:SledTestClass rdf:type owl:Class.
ex:SledTestConcept rdf:type skos:Concept.
ex:SledTestClass ex:correspondingConcept ex:SledTestConcept.
```

Listing 4.8: Linking SKOS and OWL via OWL annotation property.

Serendipitously, a third way of solving this problem emerged: the W3C SKOS Working group released a "*OWL DL Prune*" version[8] of SKOS which allows a clean integration of SKOS into OWL 2 ontologies without losing the decidability. We applied SKOS to annotate ontologies expressing business knowledge, i.e., methods and policies, like regulations (Omrane, Nazarenko, Rosina, et al. 2011). The domain knowledge encoded in the ontology provides a CV which supports KEs in the formalization of BRs that are described in written policies. In this case, we have been formalizing rules based on a European regulation (*ECE R16* (United Nations Economic Commision for Europe 2009)) concerning seat belts which is provided in many European languages. Since the working language at Audi is German, we applied labels in German as well as in English. Such documents are consulted by Original Equipment Manufacturer (OEM) as well as supplier engineers in order to fulfill the world-wide, national, regional, group and brand requirements. They describe tests, rules and instructions concerning technical products which have to be complied to.
In the next chapter, we describe how such documents can be processed in order to obtain formalized rules and ontologies. In this section, we just refer to the parts of the paper (Omrane, Nazarenko, Rosina, et al. 2011) that are important for our CV.
The DEs usually have to manage a large bulk of knowledge which is seldom entirely formally described. Therefore, it is reasonable to add informal information to concepts when it is available, especially if they have implicit or ambiguous la-

[8] http://www.w3.org/TR/skos-reference/skos-owl1-dl.rdf; retrieved 08/04/2014

bels. Legal documents, such as regulations and policies often come along with a precise definition of their terminology. For our CV, we extract these definitions from the documents and associate them to their respective SKOS concept using the `<skos:definition>` expression as shown in our case study in section 7.2. Due to this definition, the user knows exactly what the concept is about and can also use the ontological representation as a reference work for their domain's terminology.
CVs provide a key to how today's enterprise challenges, like becoming more agile and the involvement of a lot of different information, can be resolved. "Used poorly, they will be rejected as just another meaningless set of rules to follow. Used correctly, a [CV] can mediate communication between parties" (Wood 2010).

4.7. Conclusion

In this chapter, we have presented a method framework based on SWTs. Therefore, we initially defined the various used terms, like "method" or "methodology" and demarcated them from similar concepts, like "processes". The literature analysis helped to identify the concepts and their relations for our core method ontology. The resulting ontologies and rules help companies in structuring the prevailing information chaos, because this framework's purpose is to integrate not only method knowledge, but acts as a KB that incorporates the dependencies and coherences between processes, resources, guidelines and of course working methods.
Furthermore, we illustrated how this core method ontology can be supplemented with enterprise knowledge, like process or resource information that can originate from either databases, existing ontologies or other SW KBs using rules and query languages, like SPARQL. We primarily focused on extending the core ontology with metrics and linguistic information, i.e., a CV, in order create a model that can answer and fulfill our mentioned requirements in the chapter's introduction.
Introducing a maturity model along with other important method metrics and quality attributes can lead to a number of benefits for a company. For a start, the own strengths and weaknesses in the method landscape can be identified and thus, actions to improve can be taken. This allows us to predicate statements

on expected qualities, costs, time consumption or other KPIs which is a strong motive for DEs when selecting an appropriate method best suited for the task at hand. That is necessary, because today's method selection is frequently based on a method's popularity and not on a real analysis of an enterprise's requirements (*cf.* Ernzer and Birkhofer 2002). Therefore, it also serves as a tool that can be applied for the documentation of quality.
Furthermore, the company is able to stipulate minimum requirements for the method application. Hence, the introduction of method metrics models is a vehicle that can lead to a higher grade of overall quality in the enterprise.
However, the application of metrics, maturity models and the introduction of a process management which automatically comes along is expensive. The documentation, the execution and the maintenance of these assessments can be complex and is wedded to effort. Only big enterprises can make the necessary employees, that concentrate on this task, available (Jacobs 2014).
The developed method model enables a considerable knowledge increase for companies by describing its relationships, semantics and rules. Not alone by returning results for specific queries, but it also enables a straightforward visualization of complex interdependencies. It allows a complete overview of the enterprise's method landscape and thereby supports stakeholders in method selection, planning and monitoring. Furthermore, new employees can come to grips with new topics and projects more easily.
Nevertheless, there is still room for improvements. We are using direct object properties for connecting methods and thus are able to construct complex coherences, like method chains, alliances or any other aggregations. However, we did not consider even more sophisticated constructs like parallel executions by introducing gates, joins and the like, we know from process models. A meta model for such a model can be integrated analogous to the process meta model in (Eisenbarth 2013), though.
Most of the business benefits and a solution to the remaining requirements from this chapter's introduction (section 4.1) will be revealed in chapter 6 where we will introduce concepts for roles, employees, departments etc. in order to integrate our method meta model with an EAM meta model. There, we will also focus on method analysis, selection and comparison.
Technologically, at this time, the presented solution is bleeding edge. However, new technologies and standards emerge that might be useful and promising in the future, like Semantic Information Modeling for Federation (SIMF), a stan-

dard currently in progress by Object Management Group (OMG), which states to overcome OWL weaknesses in wide-scale data federation (Casanave 2012). This standard is also using a semantic approach with the intent to overcome the current data integration problems when using different authorities, technologies, formats, viewpoints and terminologies which facilitates linking enterprise data even more.

"The Semantic Web isn't inherently complex. The Semantic Web language, at its heart, is very, very simple. It's just about the relationships between things."

Tim Berners-Lee

5

Methodology

5.1. Introduction

In basics section 2.5, we concisely introduced the ONTORULE methodology. This approach serves as our foundation, i.e., Knowledge Acquisition (KA) and general Knowledge Management (KM) procedures for Semantic Web Technologies (SWTs), like ontologies, queries and rules, but we customize and extend it in order to fit our methodology, in particular our case studies. Thereby, we introduce new steps, such as a reconcilement phase, the populating of our meta models and the integration of existing Knowledge Bases (KBs). Besides, this methodology supports various stakeholders, like Domain Experts (DEs), for instance, methods engineers, managers, project managers or Knowledge Engineers (KEs) like enterprise architects, that can access the provided knowledge in their preferred view.

5.2. Approach

A more precise activity diagram than introduced in the basics section for the realization of our work is shown in Figure 5.1. This diagram, extended with additional applied and conceptualized methodological approaches, is explained briefly in the remainder of this chapter.

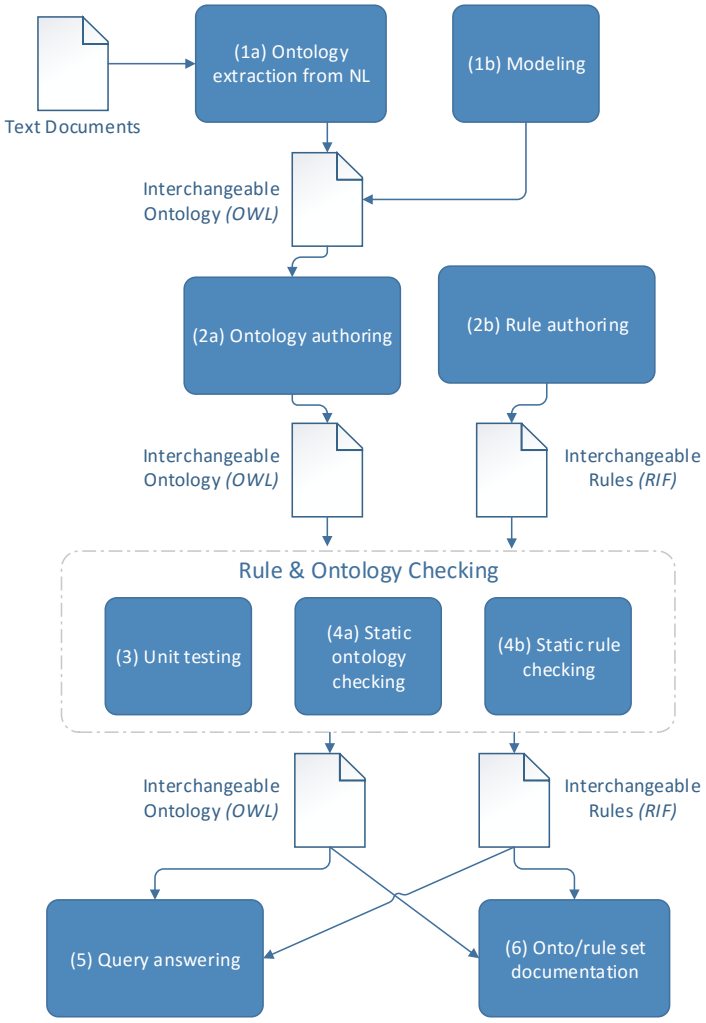


Figure 5.1.: Relevant part of the ONTORULE platform for our business case. Based on Berrueta et al. (2011).

Next to the activates illustrated in the diagram of Figure 5.1, the methodology includes support for improving the usability of the resulting application and development steps (Bonis and Bellino 2011).

Knowledge Acquisition As mentioned in section 2.1, KA is one of the first steps when performing KM.
New business knowledge emerges from various DEs, Knowledge Owners (KOs), in the Research & Development (R&D) departments and from external sources, but formal methodologies and technologies to conserve and reuse this knowledge are often unknown. The most widespread approach is to use standard office software to capture knowledge. The reason for this is on the one hand the availability of tools for creating spreadsheets or drawing diagrams in presentation software and on the other hand the good usability or habitualness. These tools are appropriate as a basis for presenting and discussing new ideas and existing knowledge. However, when the coherences between the various business objects and hence their interweaving become too complex, they are not assessable anymore.
Accordingly, the first step as a KE, when getting the assignment to capture knowledge or even better, based on an implemented process, is to get in contact with the KO and consequently developing a basic understanding of the problem domain and discussing the feasibility, requirements and further planning. For example, the KE can record what kind of questions the anticipated knowledge model should be able to answer, together with exemplary query responses.
When they have agreed upon a realization strategy, the KE begins analyzing already existing documents or presentations and performs first interviews. The newly gained insight should be captured in a first version of the conceptual model which can be implemented in an informal way which is to be translated into Web Ontology Language (OWL) later on. Depending on the kind of knowledge and application in mind, the KE should choose a fitting OWL profile. For example, OWL 2 QL when working with lots of instance data or OWL 2 RL when using rules and scalable reasoning is needed (Motik, Grau, et al. 2009).
This includes the vocabulary, coherences between classes, rules and queries of the anticipated application. Since the model is not mutually agreed upon yet, every class, relation and rule is marked as such.
Besides, methodologies on developing ontologies and KBs in general exist plentifully and should be considered (Grüninger and Fox 1995; Uschold et al. 1998; Schreiber, Akkermans, et al. 2000; Noy and McGuinness 2000; Corcho et al. 2005; Sure, Staab, and Rudi Studer 2009; Verma and Kass 2010; Möller 2012).
Corresponding, we have applied Natural Language Processing (NLP) techniques in order to extract candidate rules and candidate ontological concepts (*termino-concepts*) (Omrane, Nazarenko, Rosina, et al. 2011; Nazarenko et al. 2012) from

regulations (Figure 5.1 *(1a)*). This approach speeds up the development of ontologies, because a rough structure can be created semi-automatically, based on existing policies, laws or internal documents describing the domain. By keeping the (back-)links between the entities in the document source and the target ontological concepts by the intermediate use of *termino-concepts*, changes and updates are traceable and therefore support the ontology maintenance.
A similar, complementary approach is *CogNIAM*-based modeling methodology *Conceptual (Enterprise) Modeling* (Nijssen et al. 2012) in order to specify, validate and verify our ontologies that represent the business concepts and their relations (Figure 5.1 *(1b)*). This activity is an expert-aided modeling approach which includes interviewing DEs to gather explicit and implicit knowledge, taking all possible textual sources into account and validating their statements with a structured dialogue. On the one hand, the desired outcome of the KA activity is a "formal, complete and understandable business model" (ibid.) which can be expressed in Semantics of Business Vocabulary and Business Rules (SBVR). A great benefit of using SBVR/SE (Structured English) or any arbitrary formal Controlled Natural Language (CNL) is their readability by humans and processability by machines. On the other hand, (a subset of) SBVR models can be transformed into Unified Modeling Language (UML) or OWL (Bulles et al. 2009). Although, we evaluated SBVR by expressing our business model using this standard, we omitted its use in order to create OWL ontologies directly as illustrated in Figure 5.1 in order to concentrate our endeavor on the mentioned Semantic Web (SW) standards. The interoperability and information sharing, i.e., the federation, between various Object Management Group (OMG) and World Wide Web Consortium (W3C) standards, including OWL and SBVR, is still an in-progress standards process within OMG, namely Semantic Information Modeling for Federation (SIMF) (Casanave 2012).

Reconcilement Whichever KA approach we follow or combine – the newly gained insights should be mapped to, compared to and confronted with the already existing meta models and models in order to validate and verify them, together with the involved stakeholders.
During this reconcilement phase, every entity of the model is discussed one-by-one on its own and in coherence with the rest of the model and marked as concerted when agreed upon. This includes a definition of the classes and descriptions of the business rules (BRs). During this process new model entities

and rules can be created and existing ones can be modified or discarded. Status changes that consist of the interviewer's and interviewee's name, time and date are protocolled to trace the discussion for possible future evolution cycles. Protocolling the KO's role also acts as an indicator for the creation of views for the various user roles of this conceptual model. Adding example instances into the model helps better understanding the domain. This phase should be performed with all relevant stakeholders until ultimately a completely mutually concerted model has been developed.
Discussing the relations and class definitions together with a KO that is untrained in working with ontologies in textual form is hardly possible. Therefore, a graphical notation that represents the OWL semantics is necessary. UML being the standard graphical notation for representing knowledge about software and systems is widely known and offers many patterns that can be re-used for expressing OWL ontologies (Kendall et al. 2009; Barzdins et al. 2010; Zedlitz et al. 2012).
However, that graphical notation should be as simple as necessary to discuss the individual entities and their coherences with the KO. Therefore, the model should be expressed only with key elements, like classes and relations. More sophisticated elements, like disjunctions or complex class expressions should be supported but not visualized. These "complex" elements should be hidden until needed for the discussion.

Authoring The next step according to the ONTORULE platform is the authoring of candidate rules and ontologies (Figure 5.1 *(2a,2b)*). We performed this task in different stages – we optimized our OWL ontologies, transformed them into ObjectLogic and created the required rules concurrently while testing the desired output with constraints that checked the desired behavior. Parts of the rules could be transformed into Rule Interchange Format (RIF). However, the language had not been powerful enough in order to model our whole business logic and tool support was insufficient at this point of our development. Further tests and checks ensured the correctness of our model *(3,4a,4b)*.

Populating After successfully creating an ontological model and rules that fulfill our requirements, we want to enrich our Terminological box (TBox) with data. Therefore, we need further methods that help us to acquire data from various sources. On the one hand, this data comes from existing Information Technology

(IT) systems, for instance databases. Consequently, we reused and adapted this already formalized data for our SWT-based solution. On the other hand, since a part of our to be modeled domain is unique and has never before been modeled, we have a lot of knowledge that is not available in any formalized fashion. For this reason, we apply and adapt techniques to help us extract and convert knowledge from documents and spreadsheets and even knowledge that has not yet been set down in a written text, i.e., it solely exists in the heads of DEs.

Query answering and documentation The final activities shown in Figure 5.1 are query answering (Figure 5.1 *(5)*) and the creation of a documentation (Figure 5.1 *(6)*). The latter could be done automatically by analyzing the rules and ontologies structures and utilizing the labels and comments provided by the standards, for instance, *rdfs:comment* or *rdfs:label*. Query answering is our direct link to our application. Our main endeavor is to transfer as much of the business logic as possible into our queries in order to have an application that does only provide an interface to the user. Thereby, we can respect the different lifecycles of code, business logic and domain knowledge (*cf.* Figure 1.3) and the right roles were able to maintain their relevant parts.

Methodology Overview Figure 5.2 illustrates the methodology used throughout this thesis. Many activities resemble the ones that have been introduced in the ONTORULE methodology. However, we expanded and tailored the methodology in order to include the exploitation of existing data sources, such as databases and already available ontologies as explained in chapter 4. Furthermore, we customized the involved user roles, which is further described in section 6.3.2. Another extension compared to the ONTORULE platform is the considered recurring maintenance that is an essential part in KB and application lifecycles. Besides, the proposed languages have been partially substituted in order to comply to the SW Stack, i.e., we chiefly focus on OWL and SPARQL.
Finally, when the model is agreed upon by all relevant stakeholders, e.g., KOs, managers and KEs, it can be released by the respective roles, for instance, an Enterprise Architecture (EA) team. Either in parallel or after its release, an appropriate SW-based software solution, including the User Interface (UI) should be developed. Because our focus is not on software development with SWTs nor ontology engineering, we refer to other sources, often model-based approaches, for

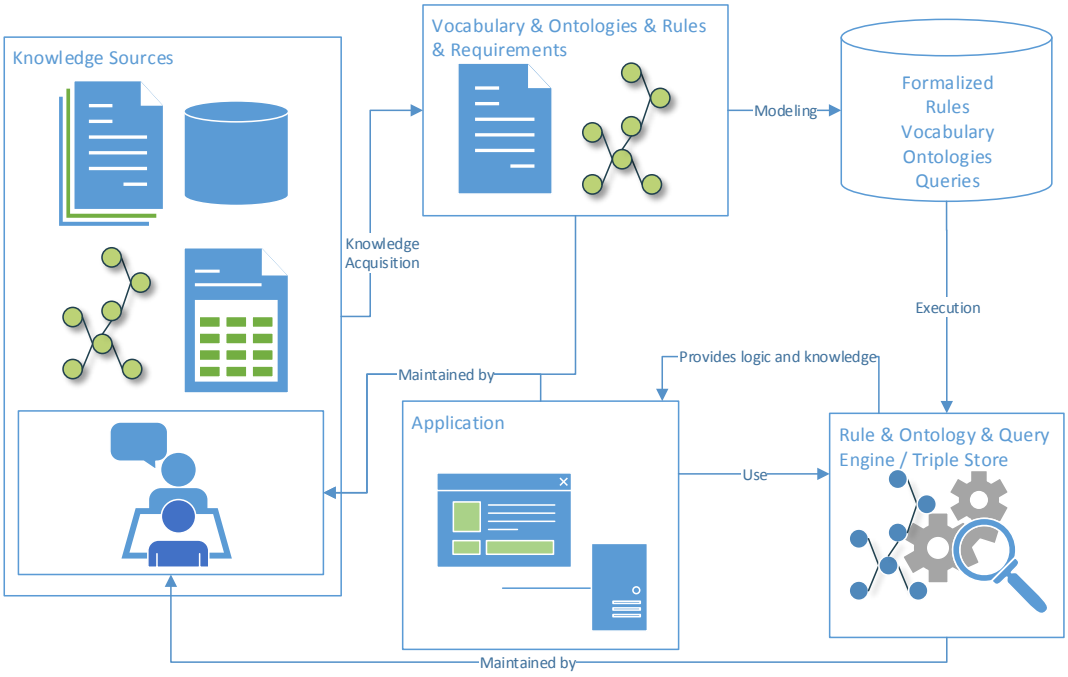


Figure 5.2.: Methodology.

further information (Brambilla, Ceri, et al. 2001; Oberhauser and Schmidt 2007; Staab and Rudi Studer 2009; Staab, Walter, et al. 2010; Farooq and Arshad 2010; Verma and Kass 2010; Parreiras et al. 2010; Brambilla, Cabot, and Wimmer 2012). Nevertheless, business and domain knowledge evolve. In order to reflect the changes, the conceptual model together with the rules and queries need to be monitored, maintained (*cf.* Figure 5.2) and updated as required, preferably by an established governance process.
Furthermore, appropriate tests, also regarding correctness, validation and performance, should be implemented and executed during the whole methodology.

5.3. Conclusion

In this chapter, we concisely presented the methodology that has been developed throughout this thesis in order to create SWT-based applications. One of the first steps is the KA, where KOs and KEs work together in order to gather and formalize existing enterprise knowledge, not necessarily using SWTs. After a reconcile-

ment phase, during which the gathered knowledge is agreed upon, tailored and improved, it is going to be formalized in rules and ontologies in the authoring phase. The following steps include populating the ontologies with data, creating queries and documentation of the models. When the models are ready for release, appropriate SW-based software solution have to be developed and connected. This step, though, is not part of this thesis.
Finally, the models have to kept up-to-date constantly in order to reflect the ever-changing company knowledge. The separation of business logic, domain and business knowledge simplifies the maintenance of these individual models, i.e., code, ontologies and rules.

"Progress depends on the exchange of knowledge."

Albert Einstein (1879 – 1955)

6

Enterprise Architecture Integration

6.1. Introduction

The main objective of this integration has been a combination of method knowledge with an Enterprise Architecture (EA) in a product development division, while keeping both Knowledge Bases (KBs) separated. This way, the input of strategic, business and Information Technology (IT) decisions on methods and vice versa can be easily inferred. As a consequence, the knowledge is exchanged and shared, but can be maintained independently.

Integrating our method meta model into an Enterprise Architecture Framework (EAF) bears many advantages. First of all, the combined meta models allow stakeholders to perform novel kinds of analyses, like impact analyses or discovering business, IT and method relations which in turn leads to a higher transparency. For example, the concern "Which system/application supports which methods?" or the responsibility of the modeled actors and roles can be identified. Furthermore, experts in method development gain profit from this integration, because the methods development can be aligned with the company's strategic goals.

Aligning methods and their context information with EA knowledge can be used to create novel kinds of Key Performance Indicators (KPIs), thus supporting man-

aging decisions, for example, by integrating the company's methods quality and maturity into statements about their overall performance and business capability. Furthermore, a company benefits from such a mapping approach through a concerted and defined meaning of the modeled concepts and vocabulary. Web Ontology Language (OWL), especially the extension Simple Knowledge Organization System (SKOS), supports the use of various labels, hence different vocabularies can be attached to the concepts, if required.

6.2. Integration of the Method Model

As a prerequisite for the integration of our method ontologies into an EA model, we first need to obtain a formalized model. We have decided to use The Open Group Architecture Framework (TOGAF) (The Open Group 2011) and ArchiMate (The Open Group 2013) as a foundation for our EA model because of their popularity and maturity (Cameron and Mcmillan 2013). However, the Open Group does not provide a formalized model of their framework but considers TOGAF as an approach that should be further refined and implemented. Nevertheless, using ontologies and other Semantic Web Technologies (SWTs) for the realization of EAs has been done for years now (Uschold et al. 1998; Dietz 2006; Sunkle, Kulkarni, and Roychoudhury 2013; Ortmann et al. 2014).
A lot of them cover the TOGAF Content Metamodel which should assist to generate EAs "that are clear and unambiguous" (Gerber, Kotzé, and Van der Merwe 2010). This formalization, for example, is suitable for the use cases introduced in their work. It does not provide the anticipated abstraction level we need for our formalization, though.

6.2.1. EA Ontology

As a consequence and inspired by the above mentioned EA frameworks and meta models, we also created our own hybrid EA ontology, based on the TOGAF Core Content Metamodel and ArchiMate. The resulting ontology is depicted in Figure 6.1, representing the known concepts from TOGAF and ArchiMate.

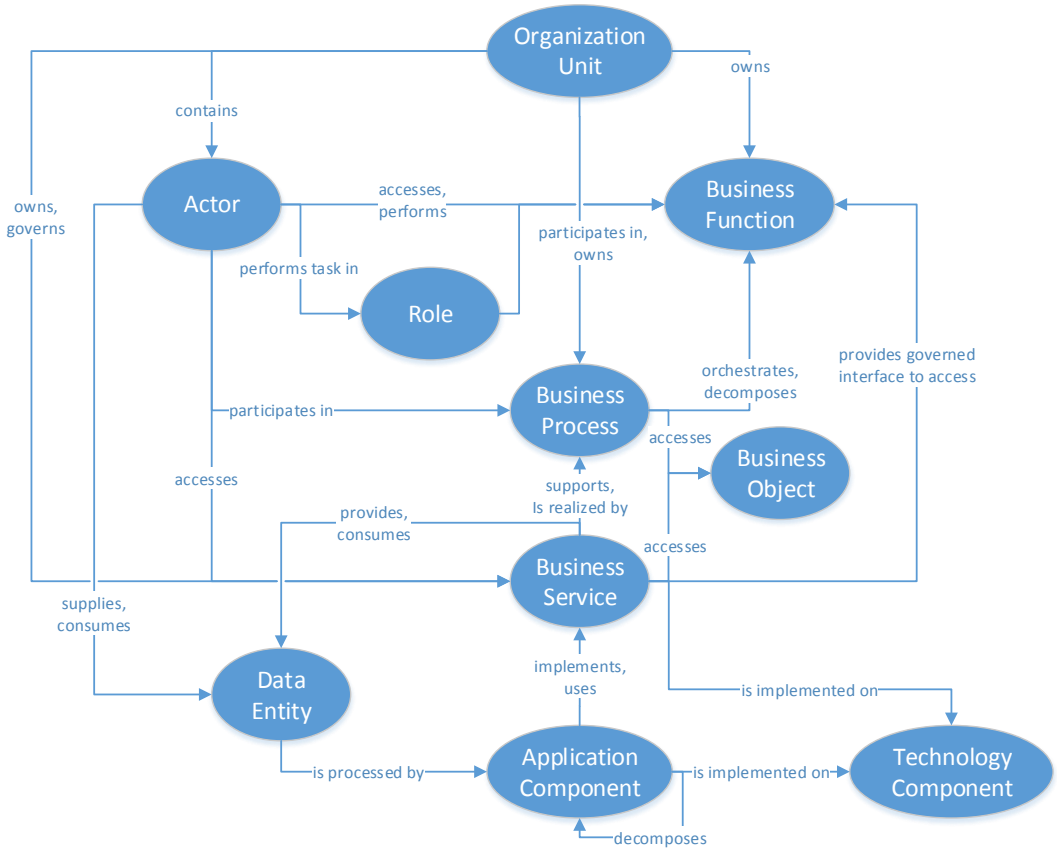


Figure 6.1.: EA ontology based on TOGAF and ArchiMate meta models.

Actor: The class of persons, systems and other agents that interact with the EA.

Organization Unit: A self-contained group of resources.

Role that an actor assumes during a task.

Business Function "delivers business capabilities" (The Open Group 2011).

Business Process represents a set of connected process actions that delivers a valuable outcome (*cf.* section 4.2.2).

Business Object represents a business entity, e.g., an invoice. It is applied during the execution of a business process (Buckl et al. 2008).

Business Service , for example, of an IT service, that supports the realization of a business process.

Data Entity represents "an encapsulation of data" (The Open Group 2011), defined and recognized by a Domain Expert (DE) as an individual of this class.

Application Component represents "an encapsulation of application functionality" (ibid.).

Technology Component represents "an encapsulation of technology infrastructure" (ibid.).

Since our method ontology has been realized using OWL, we also applied this language when designing the TOGAF ontology. An important requirement for this EA ontology has been a suitable coverage of the concepts known from our core method ontology, together with the extended method ontologies. Thereby, we want to make use of newly generated relations, for instance, from methods to business objects or capabilities. The TOGAF Core Content Metamodel, combined with the ArchiMate Design approach, fulfills this requirement and can be supplemented when specialized concepts are needed as illustrated in Figure 6.2. We use the same generic concepts (OWL meta model) and extend the EA concepts with a more specific domain meta model.

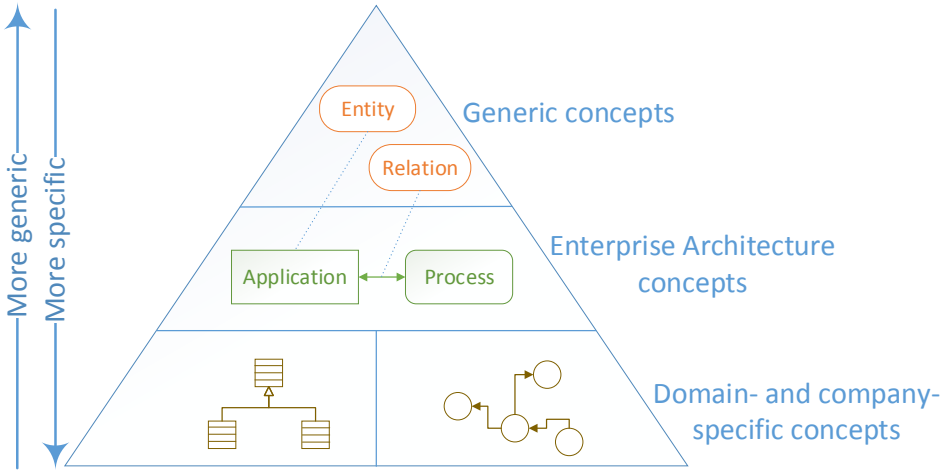


Figure 6.2.: Meta models at different specificity levels. Based on The Open Group (2013).

After the selection of this EA model, we need to establish mappings from our method ontologies' concepts and entities to the already existing elements in the EA.

6.2.2. Mapping Techniques for the Integration of the EA and Method Ontologies

Technically, we could pursue different mapping techniques for combining our ontologies: we could use the other ontology's concept Uniform Resource Identifiers (URIs) directly, which is the standard way in Semantic Web (SW). For example, by

importing the ontology into ours, we can use *ea:BusinessProcess* as a replacement for our *method:Process* in the method ontology. However, a fundamental idea behind our integration is the retained separation of both KBs, because each domain – the method knowledge and the EA knowledge – is the particular responsibility of a dedicated organizational unit and hence, changes in the other ontology can be opaque. An alternative option would be to use an upper ontology, like UMBEL (Giasson and Bergman 2015), which is certainly a reasonable choice when combining lots of domains. We do not need such an explosion of our domain for the scenario at hand, though. The technique we have chosen to map the Terminological boxs (TBoxes) is the use of an own mapping ontology for combining the various concepts, for instance, by using *owl:equivalentClass* or *rdfs:subClassOf*. This option can be implemented very fast, the mappings are traceable in one ontology, and it offers a good overview for a manageable amount of concepts, which is sufficient for the prototypical introduction.
Furthermore, next to matching the ontologies' TBoxes, we make statements about the individuals in the Assertional boxs (ABoxes) that represent our model. The respective individuals, e.g., the modeled processes, are usually matched using OWL notation, as in *owl:sameIndividual*. However, it is often the case, that two individuals are only nearly exactly the same. Therefore, the use of a 'Similarity Ontology', along with object properties like `so:identical`, is a good idea. It allows expressing different levels of identity (Halpin et al. 2010). Additionally, we use hierarchical, equivalence, part-of and other arbitrary relations.
The main objective of this integration has been a combination of method knowledge with an Enterprise Architecture Management (EAM) in the product development division. A fundamental idea behind the integration is the retained separation of both KBs, because each domain – the method knowledge and the EA knowledge – are the responsibility of different departments.
This way, the input of strategic, business and IT decisions on methods and vice versa can be easily inferred, while the knowledge can be maintained independently. Thereby, we do not just integrate methods, but also method projects, i.e., projects that develop or improve new and existing methods. The list of concepts and relations needed for this mappings have been directly or indirectly derived from our requirements, resp. the queries for answering them, mentioned in sections 1.2 and 4.1.
Since our method ontology has been realized using OWL, we also applied this language when designing the TOGAF ontology.

6.2.3. Mapping the EA and Core Method Ontologies

When comparing our core method ontology's with our EA ontology's concept names, we encounter some obvious similarities which are candidates for concepts intersections. An equal or similar name, however, does not automatically infer synonymous semantics. The considered key concept `Process` from our core method ontology and the concept `Business Process` from the EA ontology are identical, though, especially when regarding the TOGAF "Process Modeling Extension". It states that "a process represents flow of control between or within functions and/or services" (The Open Group 2011) and a "sequence of activities that together achieve a specified outcome" (ibid.). In this process definition, we encounter the two terms *function* and *service* – the remaining terms have already been sufficiently considered or their meaning is trivial or self-explaining. The first term *function* has already been regarded in our method definition, but we still have to investigate if they mean the same. The second term *service* is something entirely new, that we did not yet consider in our method model, but which is present in the EA ontology as the concept `Business Service`. Furthermore, a process is related to an outcome `Product` in both definitions.
The conceptual mapping between both ontologies is depicted in Figure 6.3. We state that both concepts are equal even though the different processes can vary in their granularity.

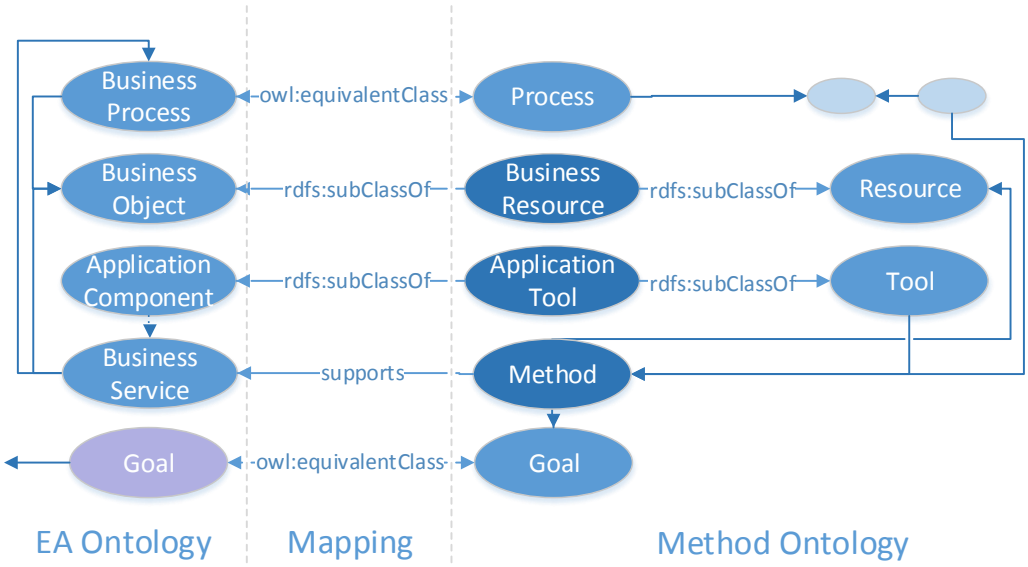


Figure 6.3.: Mapping of EA and core method ontology.

Our intentions for the methods' input and output Resources are roughly semantically covered by the concept Business Object, known from the ArchiMate Business Layer Metamodel. However, the method's resources as defined in section 4.6.1 span a much wider range of possible method in- and outputs, for instance, people, roles or data. Because these other in- and output types are partially present as discrete classes in the EA ontology, only a subset of our Resources are mapped to a Business Object. Processes can perform all kind of Create, Read, Update, Delete (CRUD) operations on a business object, which either represents a virtual, for instance, an information or data object, or a physical object (or both) (The Open Group 2013). This applies to our methods' relations and the corresponding resources, as well. Nevertheless, only a subset of the modeled EA business objects are pertinent for the analysis of our core method ontology and vice versa. Therefore, both concepts share the common subclass Business Resource (*cf.* Figure 6.3). This subclass is modeled in a domain ontology (DO), importing the upper method ontology.
The same reasoning applies to the concepts Tool and Application Component. A lot of business software is of no interest for the method domain and tools of the method domain can also include physical devices and implements. Consequently, we introduce the concept Application Tool.
The various individuals in the ontologies' ABoxes are then mapped to their counterpart using properties that express the appropriate similarity.
Additionally, Methods represent a link between Business Services, Process-Actions and Application Components/Tools. They express, how and when a Business Service can be applied to a specific process.
The final depicted mapping between both ontologies deals with the motivation extension, since the utilization of methods as well as stakeholders in EA intend to achieve a Goal. Albeit, we have to take into account that these goals may represent a different level of granularity.
The remaining concepts of the core meta method model feature no counterpart in the EA ontology, even though both models feature a concept named Function. A function in TOGAF "delivers business capabilities closely aligned to an organization [...]. Also referred to as 'business function'" (The Open Group 2011), whereas a function in our meta method model represents an appropriated behavior of a technical system (Weigt 2008). This kind of technical function, for instance, a vehicle function, is also called *feature* in the technical context. The combination of design method knowledge with other business knowledge ob-

jects, e.g., functions, properties or solution concepts, will be evaluated in section 7.5.

6.2.4. Further Mappings

In the previous section, we presented how to map our core method ontology with a core EA meta model. However, we have also introduced a metrics and a resource ontology in chapter 4, along with introduction of a Controlled Vocabulary (CV). These specific domains can be mapped as well and are the basis for the generation of additional analyses which benefit the company. The conceptual mapping is presented in the following subsections. The possible analyses and emerging artifacts based on these mappings are presented in section 6.4.3.

Metrics mappings

TOGAF does not yet include a maturity model that could be used to measure its adoption. The standard does include a chapter about other Capability Maturity Models (CMMs), i.e., Capability Maturity Model Integration (CMMI) and the US Department of Commerce (DoC) Architecture Capability Maturity Model (ACMM), though (The Open Group 2011, Chapter 51). The ACMM can be used to calculate a maturity rating for the overall EA in a company, divided in six levels. These levels[1] resemble the levels introduced in section 4.4.1. These CMMs can be used to measure one's architecture in order to gain a higher assessment ranking. Certainly, this is also possible for the management of methods in an EA. However, our introduced metrics ontology is not meant to assess an architecture in its entirety, but to assess individual methods and thus make them comparable. Nevertheless, if all the modeled methods are provided with the relevant KPIs, the architects or methods engineers can of course deduce an overall maturity for the whole method landscape.
Another way of mapping our metrics ontology with the TOGAF would be the introduction of EA patterns, "although architecture patterns have not (as yet) been integrated into TOGAF" (ibid., Chapter 25). Patterns are defined as "an idea that has been useful in one practical context and will probably be useful in others"

[1] 0: *None*; 1: *Initial*; 2: *Under Development*; 3: *Defined*; 4: *Managed*; 5: *Measured*

(Fowler 1996), so they have much in common with methods that are reusable and shall be measurable with metrics. As in architecture patterns, we use the "ilities", called "forces" in the pattern terminology, for the assessment. Some examples for these forces are the robustness, manageability, repeatability, scalability, ease-of-construction or ease-of-use which are mostly congruent to the examples given in the metrics ontology (*cf.* section 4.4.3). However, since they are not yet integrated, we have to choose another solution for the mapping.
Because of the lack of a tighter mapping possibility, we do not link the method metrics directly with the EA model. The metrics ontology is mapped as explained in section 4.4.3. Nevertheless, artifacts based on method "ilities" can be generated by appropriate queries as seen in the following section 6.4.3.

Resource mappings

As explained before, a method's `Resource` resembles a `Business Object` of the EA ontology. However, the resource ontology introduced in section 4.6.1 features more and specialized classes in the resource's context, for instance, `Products`, `Concrete Resources` and `Abstract Resources`, analogous to the method's concrete and abstract versions.

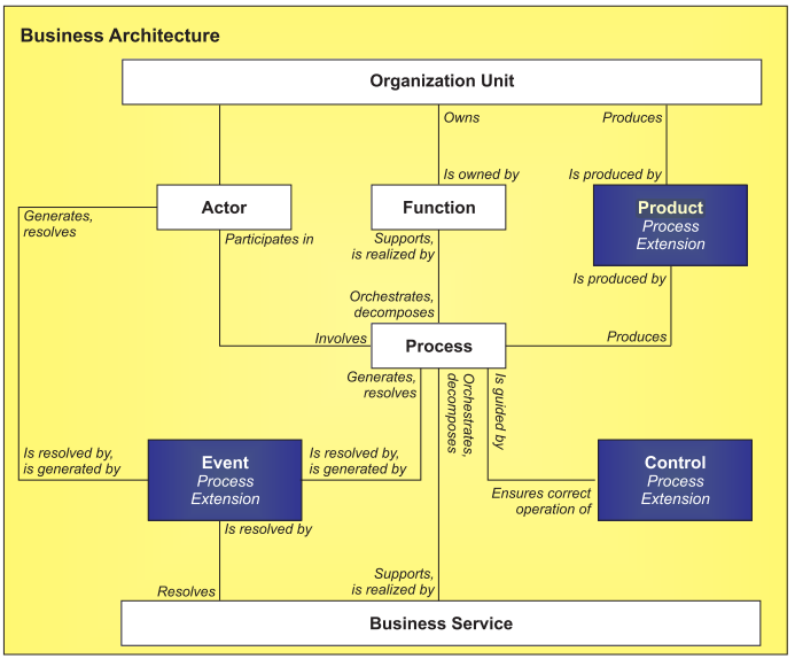


Figure 6.4.: The process modeling extension (The Open Group 2011).

TOGAF features a process modeling extension, depicted in Figure 6.4, that introduces the concept *Product* (among others) to the EA meta model that links the concepts *Organization* and *Process*, in our case `Organization Unit` and `Business Process`. The `Product` of our resource ontology can be directly mapped to this *ea:Product* using *owl:equivalentClass*.
Further mappings are not necessary, because we do not wish to merge the domains as strongly as possible, but keep them separated. In fact, the TOGAF standard mentions, that typically, EA does not deeply concern itself with process flows which are introduced by this extension and hence we do not drill into this topic, either.

6.3. Views and Roles

6.3.1. Introduction

Our evaluation revealed, that because of the numerous roles that our end users represent, it is no easy task to create one application or view that fits them all. Therefore, we decided to develop individualized views and artifacts for the diverse stakeholders.
When working with comprehensive knowledge models, specific user roles and application often require only parts of the broad knowledge that is covered by the model. In the case of ontologies this means that various users focus on different portions of an ontology (Zuo 2006). Being even more restrictive, agents may only be allowed to see and work with an approved subset of the knowledge, i.e., the *principle of minimal authority* (Denning 1976) can be incorporated. Furthermore, the offered information can be edited in order to present the target audience's vocabulary, the knowledge can be aggregated or shown in a fine-grained level of detail. In general, information usage (of information entities) may vary with respect to: different levels of *granularity*, different *vocabularies*, different *scopes of a domain*, different *contexts of use* and different *perspectives* (Zuo 2006). The "perspective of interest from which an expert examines the [KB]" is called the *viewpoint* (Marino, Rechenmann, and Uvietta 1990). This emphasizes two different aspects: the *focus* and the *view angle* (Ribière and Dieng-Kuntz 2002), as illuminated in Figure 6.5.
Viewpoints can be classified into two different kinds: the perspective (consensus

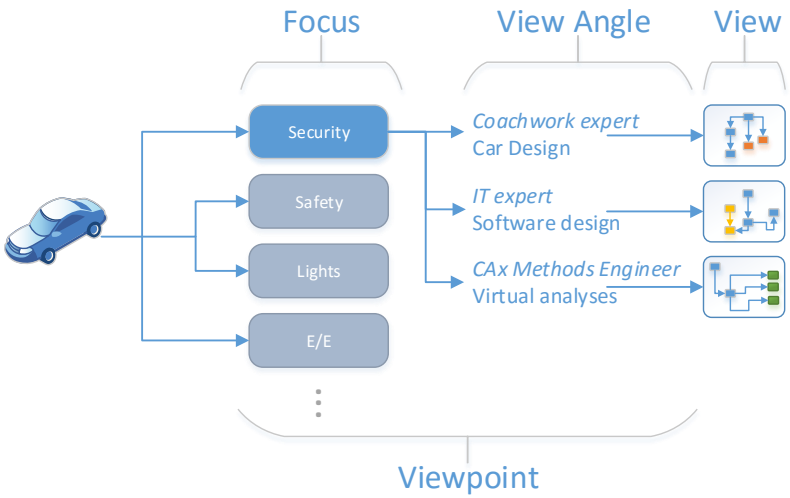


Figure 6.5.: An example of multiple viewpoints for a car's description. Based on Ribière and Dieng-Kuntz (2002).

of experts) and an opinion (non-consensus) (ibid.). On the one hand, in the figurative sense, a perspective is represented by a commonly approved ontology that reflects widely accepted knowledge. A common vocabulary consists of common knowledge (global) views by at least two different viewpoints. Common knowledge may be a global concept or a bridge rule between two (or more) partial descriptions (Djellal, Hemam, and Boufaida 2010). On the other hand, experts can create or modify ontologies that reflect their own understanding of a particular domain.

Definition 6.1 (*Architecture view*)

"Architecture views are selected parts of one or more models representing a complete system architecture, focusing on those aspects that address the concerns of one or more stakeholders" (The Open Group 2011)

In our case, we have to differentiate between architecture views for stakeholders of the EA in general and the necessary views for the roles associated with the methods domain.
Therefore, we will introduce the corresponding roles for our methods ontology and matching parts of EAM, based on stakeholders identified during the ONTORULE project (Bonis and Bellino 2011) and our industrial case studies (*cf.* chapter 7). Furthermore, this section provides the information necessary to model

these views with ontologies and the requirements and expected benefits following this endeavor, as it is a prerequisite for the realization of the method architecture artifacts provided for the various involved stakeholders.

6.3.2. Stakeholders

During the whole knowledge management lifecycle, various user roles interact differently with the KB and pursue sundry tasks and goals. These user roles can be applied to any large industry development department, but may differ slightly to the ones identified during our case studies in the industry and the roles identified during the ONTORULE project. Together with our project partners, we identified the following personae, which are relevant for the application of the ONTORULE methodology (Bonis and Bellino 2011), i.e., developing a business application that is based on ontologies and rules:

- Luis, the *IT Specialist*, for instance, an IT Architect, Software Engineer or Developer, ensures the technical maintenance and is in charge of developing the business application.
- Marc, the *Business Analyst*, e.g., a vocabulary or rule analyst, is responsible for the formalization of the business knowledge necessary to develop a business application.
- Gary, the *Senior Business Expert*, is the head of a business domain in the enterprise, knowing relevant vocabulary, policies and rules in combination with his experience, overall knowledge and vision for defining the business model.
- Alice, the *Domain Expert* (DE), knows her domain in detail and interacts with business rules (BRs).
- Joana, the *Operational* or *End User*, will use this business application in order to realize relevant business operational tasks.

Most of the roles that are connected to the identified personae have already been mentioned in previous sections and chapters. During the development of our method model, its combination with the EA domain and the following case studies, we identified very similar relevant roles, but titled them slightly differently in our context. Therefore, we will discuss each persona briefly and refer to the pertinent role.

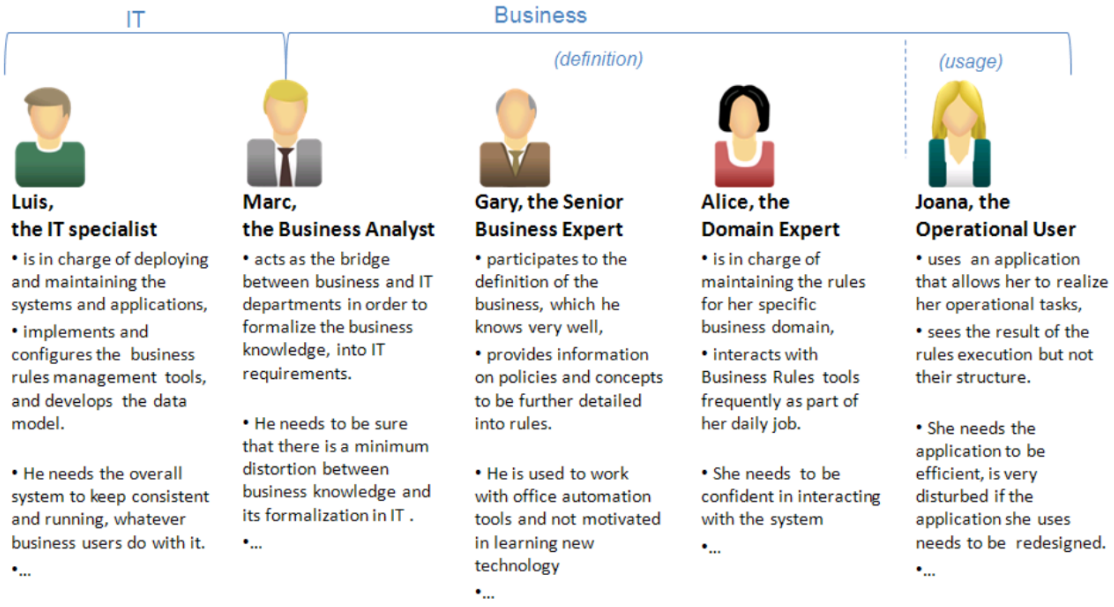


Figure 6.6.: Overview of the personae developed during the ONTORULE project (Bonis and Bellino 2011).

In the context of our previous chapters, we did not mention IT Specialists or developers, but surely this role is necessary for the development of a business application, based on KBs, as seen during our case studies in the following part of this thesis. They visualize the knowledge for the appropriate roles and enable interaction. The queries, designed by the business analysts, and the required User Interfaces (UIs) for the various roles are the important working interfaces between an IT expert and the following roles.

The business analysts is a synonym for *Knowledge Engineer* (KE) formerly introduced in this thesis. A Knowledge Engineer (KE) is a person who masters a Knowledge Acquisition (KA) methodology; a KE does not need to have knowledge of any specific domain except the generic domain (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). An actor of this role can be, for instance, an enterprise architect, an IT expert with modeling knowledge or a domain engineer that is capable of modeling. This role is very important for a SWT-based approach, because it is responsible for the correct and comprehensive development and formalization of the domain models, TBoxes, mappings, rules and queries and hence making them applicable. *DEs* want an easy to use and fast solution to monitor, view or manage their knowledge but they are not trained using ontology and rule editors. Therefore, they are supported by KEs that conserve knowledge in a formal way, e.g., with ontology and rule editors and present the knowledge in

correct models. For that reason, this role does not necessarily need knowledge, but talent and quick-wittedness in the appropriate business domain, because this role acts as the bridge between and works tightly together with domain, senior business and IT experts.
The senior business expert represents a *manager* of an organization unit or department, for example, a process owner that handles the overall consistence of knowledge. This role is experienced, knows the own organization unit's contexts, i.e., business relations to other customers, and has a much broader and more high-level viewpoint.
The DE has already been mentioned several times. Detailed knowledge of rules and vocabulary, specific analyses requests and interacting with the business processes on a deeper level affect the DE's necessary viewpoint. The DEs need a solution to formalize complex knowledge quickly and efficiently. Ideally, they manage ABox data in an existing ontology, adapt BRs to new situations and work together with KEs to perform more complex tasks.
In our running example from the automotive domain, DEs can assume various roles: vehicle DEs, e.g., a Computer Aided *x* (CAx) specialist, for instance, a methods engineer that develops new and optimizes existing methods, or policy experts that want to formalize legal documents or regulations.
A methods engineer is a role in methods engineering, not to be confused with method engineering.
Both disciplines are cousin to each other, whereas methods engineering is a kind of industrial engineering, i.e., it is concerned with the design of industrial productive processes with human participation. That means, a methods engineer supports the workers in performing their processes, generally converting raw materials to finished products, by analyzing, assessing and optimizing their tasks.
Method engineering, on the other hand, is a discipline in software development. It deals with "the design, construction and evaluation of methods, techniques and support tools for information systems development" (Brinkkemper 1996). Another basic principle of method engineering is the situational adaption and tailoring of existing methods, depending on the method's context and utilization. Frameworks, so called Computer Aided Method Engineering (CAME) tools, support the method engineer in executing his work.
The final introduced persona is Joana, representing an end user of the business application. In our case, we do not need specific end user viewpoints for this role, because the methods as well as EA model-based application end user is congru-

ent to the roles defined above, i.e., *KEs, DEs and managers*.
Further roles that can be considered are, for example, a demarcation between upper-level and lower-level management, operational managers, customers, project managers and others (The Open Group 2013).
Next to the functional requirements for a multiple user view systems (Zuo 2006), during the KA for creating the individual persona views, it is important to consider to (Ribière and Dieng-Kuntz 2002):

- express the overlapping, common parts and the differences between the roles' models.
- detect and try to solve the terminological conflicts.
- heed the various viewpoints: "several experts according to their specialty or their way to tackle the problem solving, may have divergent analyses or divergent understandings of a same object" (ibid.).

That means, that multiple experts that want to access the same data, usually need multiple viewpoints. Furthermore, single experts may need various viewpoints, as depicted in Figure 6.7.

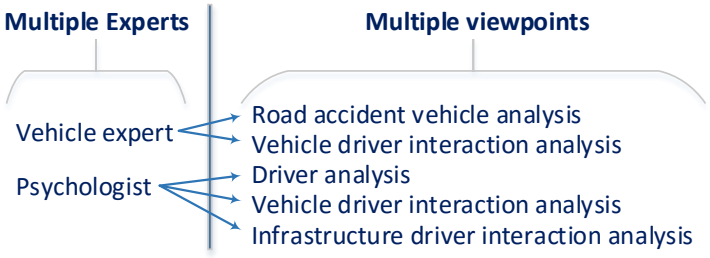


Figure 6.7.: Example of the links between multiple experts and multiple viewpoints in an application. Based on Ribière and Dieng-Kuntz (2002).

In conclusion, we can say that "the notions of multi-expertise and multi-viewpoints are closely related" (ibid.) and should be considered as such. As mentioned, this thesis' approach tackles the challenges by developing customized CVs, rules and queries.

6.3.3. Creating Views

Presenting the relevant parts of the ontologies to the designated stakeholders, while focusing on their regarded tasks is the purpose of different viewpoints. The creation of a customized view for one user role is usually achieved by a process that takes an existing, global view and tailors and adapts this view according to the user role's needs. This process "consists of a sequence of primary steps applied in original view in order to manage redundancy, inconsistencies, concept omission and addition, multiple object classifications, structure and property changes and terminology and multi-lingual support in the user view" (Zuo 2006).

The technical realization of these views can be achieved through various implementations, whereas we follow the structure and standards proposed in the SW Stack, i.e., the views are commonly tailored extracts of a KB (OWL), queried by SPARQL. When connecting databases to the system it is possible to see local ontologies that map these databases as viewpoints, which is then called a multiple ontology system. A hybrid ontology system is a variant where all these local ontologies are combined in a general ontology in an upper layer of the architecture (ibid.). Additionally, the information can be derived and enhanced by the use of a rules language, which is not necessarily the Rule Interchange Format (RIF). The information is then presented to the end user in an appropriate form, known as *artifacts*, which involves catalogs, matrices and diagrams (The Open Group 2011) among others.

First of all, the necessary subjacent ontology modules can be combined and imported using sundry techniques (Ouziri 2009) to deal with multi-viewpoints, for instance, by using inherent features of OWL, e.g., *owl:imports*, feature models (Langermeier, Rosina, et al. 2013; Langermeier, Driessen, et al. 2014) — briefly explained in section 4.6 —, $\mathcal{E}$-Connections (Grau, Parsia, and Sirin 2009), Package Based Description Logics (PDL) (Bao 2007), C-OWL (Bouquet et al. 2003), Distributed Description Logics (DDL) (Borgida and Serafini 2003) or Interface-Based modular ontology Formalism (IBF) (Ensan 2010), to name just a few. Stuckenschmidt explains the creation of viewpoints with Description Logic (DL) (Stuckenschmidt 2006): it is possible to define viewpoints by a subsumption of various relations with $\top$, hence creating a sub-vocabulary, for instance, by saying that "$Person \equiv Mother$ iff we subsume the relation *hasGender* of the class *Mother* with $\top$" (ibid.). Furthermore, in order to fulfill the different requirements of the sundry

agents regarding a big ontology, user profiles can be created (Arara and Benslimane 2004): not every user is interested in every concept, instance, object or data property. Having a multi-viewpoint ontology, it is possible to add fuzzy/imprecise/uncertain relations between the alignment of the different viewpoints (Djellal, Hemam, and Boufaida 2010) in order to create non-consensus, i.e., opinion-based, viewpoints.

Next to realizing views on the ontology tier of the SW Stack, views can also be created by upper tiers, i.e., the query tier with SPARQL and derived and corresponding languages, like *vSPARQL* (Shaw et al. 2011) or SPARQL Inferencing Notation (SPIN) (Knublauch, Hendler, and Idehen 2011), whereas the latter is a SPARQL-based rule and constraint language for the SW. SPIN provides additional technologies, for instance, SPARQL Web Pages (SWP) (Knublauch 2014), for creating template-based views for the web and for rendering SW data. These SPARQL-based languages make heavy use of SPARQL *CONSTRUCT* statements in order to generate new graphs that contain tailored data and ontology modules. In this thesis, we used several techniques in order to create customized views for the stakeholders. First of all, we used Frame Logic (F-Logic) and ObjectLogic, which is evaluated in chapter 7. Furthermore, we applied SPARQL and SPIN approaches in order to create SWT-based applications views.

Thereby, we differentiate between views, i.e., the displayed artifacts for each user role, and the underlying sets of KBs, query statements and rules. Techniques for splitting large ontologies into modules or merging or aligning smaller modules to create a more cohesive SW application has already been adequately elucidated. In addition, we experienced the need to create multiple rule sets and multiple query sets. Thereby, a representative of a relevant user group, for instance, a particular DE, can adapt and create the statements relevant to his domain without being confused by statements that are irrelevant for his focus or view angle. Next to a more comprehensive overview, multiple other benefits arise: usability increases self-evidently, new knowledge can be implemented faster and the performance of such applications can rise, when only a subset of rules is regarded. Most importantly, rules can be used to create different semantics for each user (role), and can be "changed dynamically according to usage scenarios for that user" (Zuo 2006). For example, properties, like a car's sportiness or security values, can be calculated by harnessing different KPIs or by using different units, depending on the user profile, for instance, a DE vs. a manager or an employee vs. a customer, and hence have a different outcome. Furthermore, security issues can be imple-

mented this way, allowing the user only the access to parts of the KB that he is allowed to see ("*principle of minimal authority*").

6.4. Method Analysis and Selection

In this section, we focus on the analysis, classification, selection and application of the modeled method knowledge, combined with EAM knowledge, i.e., scenarios in an enterprise, where methods are developed, analyzed, compared and applied.
Ernzer and Birkhofer have developed a model that differentiates the method selection on three levels (Ernzer and Birkhofer 2002; López-Mesa 2003). Starting from a set of chaotic, unclassified set of methods, the three resulting levels cover methods for specific scenarios:

Method pool selection: A pool of methods for instance, in databases, models etc. These are the methods that are produced and published by academia or industry. The method pool consists of domain-independent, but standardized and useful methods.

Strategic level selection: A selection of methods from the method pool that fit the company's needs.

Operational level selection: The selection of fitting methods by a user for a task at hand, for example, a method-mix that suits a project.

The methods that are modeled in our ontology and are combined with the EA fit into the strategic and operational level, because we only consider methods, that are already in use, planned or otherwise known in the enterprise. Depending on the stakeholder, we can designate these two lower levels. On the managing level and in EAM we mostly consider the strategic level. DEs and method engineers mainly use our framework for the operational level. However, the method model introduced in chapter 4, independent on its usage in a company, can of course also be applied for the method pool level.
In the remainder of this section, we will show how our framework supports the selection on the strategic and operational levels and how the methods in our ontology can be classified and compared. Finally, we combine these insights for the creation of method artifacts, i.e., appropriate queries and views concerning the methods and in consequence, their resulting business benefits.

6.4.1. Method Classification

Classifying methods helps to find relevant entries in the ontology when either the proper name is unknown or the user wants to browse through the different categories. Of course, the selection of methods by predefined queries is also some kind of classification, which is dealt with in the next sections, though. Here, we consider the hierarchical possibilities given in ontologies by introducing super- and subclasses, as well as by adding properties in order to demarcate between the method classes.
First of all, we decided to classify the methods in our KB into the relevant CAx disciplines, *viz.*, Computer Aided Design (CAD), Computer Aided Engineering (CAE) and Computer Aided Testing (CAT). Because one of our goals is the distinction between virtual and physical methods, these two classes are at the top of this taxonomy. This structure can be extended as required by introducing classes of methods for each focus, for instance, 'security', 'safety' or 'Noise, Vibration, Harshness' (NVH) as clusters in a Digital Prototype (DP) use case.
Generally, the classification of design methods is put into the following categories (Wallace 2011):

- Algorithmic methods (describing states and transformations)
- Tactical methods (deploying methods)
- Operative methods (solving problems)
- Strategic methods (planning projects)
- Reasoning methods (processing knowledge)
- Others, including human intervention, mental actions, physical actions and resources

Additionally, a general exemplary classification of methods and tools has already been illustrated in Table 4.2 on page 102.
We decided to omit these general classifications, because our ontology is meant for a domain and purpose specific situation. Nevertheless, it can be supplemented easily if necessary and new analyses queries can be implemented in order to make use of such a structure.
López-Mesa (2003) states, that a distinction between divergent and convergent methods is also relevant in the industry, when searching for methods that help in appropriate situations: "Divergent methods are those aimed to search for solutions, whereas convergent methods imply the imposition of value judgement".

Furthermore, López-Mesa suggests a distinction of innovative and adaptive methods: "Innovative methods are those appropriate for radical new ideas that imply a high level of uncertainty, whereas adaptive methods are appropriate for the improvement of mature concepts" (López-Mesa 2003).
As we can see, the classification is a multi-dimensional challenge which strongly depends on the desired outcome, i.e., method artifacts.

6.4.2. Method Selection

The implementation of a particular algorithm for the selection of a method should be realized by the people working with the methods, supported by KEs and IT experts, because there is not one single best technique to perform the selection. Depending on the situation and desired outcome, every algorithm has its particular benefits and drawbacks. Usually, standard algorithms can be taken into consideration, such as algorithms from the disciplines of optimization, statistics, decision theory, game theory, system theory and many others (OptiV 2006). More concrete examples for method selection algorithms are Highlighting Technique (HT), Advantages-Limitations-Uniqueness-Opportunities (ALUO), Pugh method and Rating & Weighting Method (R&WM) (López-Mesa 2003) or multi-criteria decision algorithms like the Weighted sum model (WSM) or Analytic Hierarchy Process (AHP). However, in a customized method and EA model, it will often be necessary to develop new or adapt algorithms, realized in rules and queries, for solving specific problems and challenges. Another important point is, that we do not want to make the selection decisions solely based on implemented algorithms, but offer starting points to the stakeholders.
Approaches known from (Situational) Method Engineering that offer guidelines for the correct selection of methodology fragments or components (Honke 2013) can be adapted and implemented as well.
Our approach aims at providing a set of possible applicable methods for a chosen scenario - the final decision is always done manually by a method expert, because each method application is unique and the expert's knowledge, experience and intuition should always overrule. Ultimately, in most situations, different suitable design methods can be used for the same type of problem.
Retrieving potentially relevant methods for product development, based on soft query criteria and then customizing the methods to the task's and user's require-

ments is also known as the research discipline of *method adaption* (T. Braun and Lindemann 2004).
Nevertheless, T. Braun and Lindemann suggest three starting points for the method selection (T. Braun and Lindemann 2003), that are depicted in Figure 6.8. As opposed to Ernzer and Birkhofer, T. Braun and Lindemann's selection model is less abstract, because it is the basis for an actual implementation, i.e., the MAP-Tool (*cf.* chapter 3).

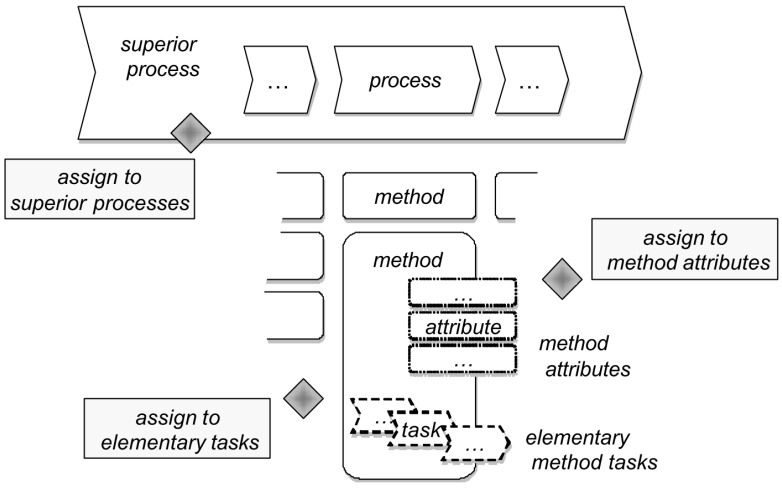


Figure 6.8.: Starting points for method selection (T. Braun and Lindemann 2003).

Assigning the considered task to an appropriate process in a superior process model is the "classic" way. One such superior process model is the Product Development Process (PDP) that is usually modeled in sundry facets concerning different departments, abstraction layers and detail levels.
A second option is to compare the task at hand's requirements, application, boundary, target and pre-conditions with the available associated method attributes.
The final possibility is to "scan the underlying task for its required basic tasks and oppose them to corresponding elementary method tasks" (ibid.).
In our case, we mainly follow the first two options, whereby, having an ontology with ideally completely linked concepts and entities, the kind of selection options blur the line. That is, we model the direct references between possible method applications and the considered process actions by introducing a `Concrete Method` between the concepts. However, our approach also features more indirect relations between the superior process and methods, by supporting queries concerning the related method's concepts and the `Business Process (Action)`, i.e., via roles, organization units, business objects or the method goals, that can be connected to business processes via the strategic level.

The second option is also supported, when it is unclear which method or even method chains are suitable for a task, by performing queries against the various method quality attributes: the related KPIs can be consulted which allows the user to select a method, e.g., based on financial information, quality aspects, time information or a combination thereof. For example, we have defined rules in our case study that calculate the duration of a method chain's application, including parameters like the method's lead time, execution time etc. All the required data for the calculations has been retrieved from the underlying ontology. If the complete duration is less than our possible time window, for instance, between two milestones in the PDP, the method chain is marked as suitable under this condition. Next to the time-constraints, we considered further methods criteria, i.e., quality, maturity and costs, whereas the latter is often the most important decision factor as stated by Reich: "Cost-effectiveness is the driving force for method utilization, but estimating cost-effectiveness is non-trivial, subjective, and context dependent" (Reich 2010).

Furthermore, the proposed methods are always dependent on the availability of their direct input resources. The method chains calculation is more complex, because the solution space is much greater when connecting single methods based on their required input and produced output.

In order to save execution time during run-time of the Business Rule Management System (BRMS), we pre-calculated some often required results by materializing the relevant rules.

In addition, our meta model supports the retrieval and selection of methods by querying the ontologies for names, labels, descriptions and other useful attributes. At this point, the use of alternative labels or descriptions can make a huge difference, because the terminology of the search parameters as well as its solution space can be adjusted to the user, for instance, by displaying the solution in the correct language.

Usually, an offered decision support for the method application makes use of the above mentioned selection options in a combined way. For example, we can propose a set of allied methods, taking into account some arbitrary quality attribute, for a specific point in the PDP, for instance, a particular milestone in German or English language.

Besides, the selection of a method can be based on its classification, as introduced in section 6.4.1.

The final option proposed by T. Braun and Lindemann, the selection by elementary tasks, means to select methods based on their underlying, atomic procedure steps. We earlier stated, that we want to consider the methods' inner workings as a black box. Albeit, the introduced method model features the class `Procedure`, which is an extension point that can be used if required for specific use cases. However, our view angle has been mostly influenced by EA and therefore, we omitted introducing elements, that are even more elemental than procedures. Nevertheless, methods are connected by various properties, like `derived_from` or `is_component_of` and the ontology can be queried by these relations in order to return hierarchical sub- or super-methods.

6.4.3. Method Architecture Artifacts

The visualization of the obtained data from queries against the KBs, customized and selected for the appropriate user role view, is realized in *method architecture artifacts*.

Definition 6.2 (*Artifact*)

"An artifact *is an architectural work product that describes an aspect of the architecture. Artifacts are generally classified as catalogs (lists of things), matrices (showing relationships between things), and diagrams (pictures of things) [...]" (The Open Group 2011).*

In this section, we will illuminate the various diagrams, catalogs, matrices and other forms of data representation that can be concluded by the combination of our method and EA KBs. Typical EA visualizations are development plans, master plan schemes, platform graphs, domain-specific models, successor diagrams, portfolio schemes, life cycle figures, roadmaps or information flow charts; the control visualizations include schemes, such as pie, bar, line or spider charts (Hanschke 2013). Further architectural views and visualizations can be found in The Open Group (2013) or as *V-Patterns* in Buckl et al. (2008).
Many of these artifacts have been evaluated in section 7.5 (Property-Driven Development (PDD) case study), including the introduced product ontology, that is ignored here, because we want to focus on the seam between EA and method knowledge and not delve into PDD/Product Lifecycle Management (PLM).

We will omit artifacts that are already known in EAM. Furthermore, we settle for the artifacts that promise a business benefit. For example, a list of `Concrete Methods` alone, without the relationship to its `Abstract Method` or the corresponding `Process Action`, does not make sense. Nevertheless, the hereafter introduced artifacts are not intended to be exhaustive and can of course be extended as required. Moreover, we confine ourselves to four stakeholder roles that have been selected in section 6.3, i.e., IT experts, business analysts/KEs (including enterprise architects), senior business experts/Managers (MGRs) and DEs (including method/CAx experts).

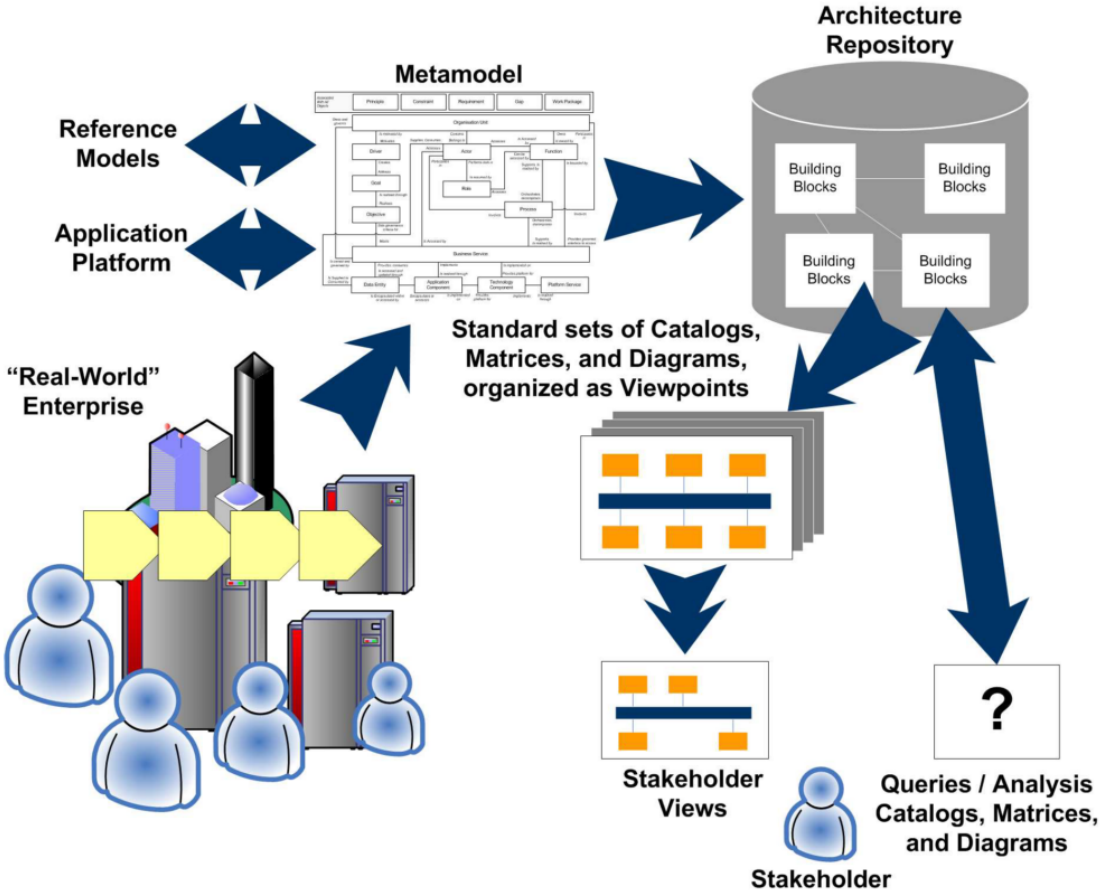


Figure 6.9.: Interactions between the meta model, building blocks, diagrams and stakeholders (The Open Group 2011).

Figure 6.9 depicts the kind of interactions between the earlier introduced components, i.e., the real enterprise which is modeled in a meta model, resp. our ontologies, mapped with connecting meta models, i.e., our method ontologies. The ontological entities, called building blocks in TOGAF, are presented to a stakeholder using the various method architecture artifacts and are the basis for extended queries and analyses.

Catalogs The most simple form of data representation is a list of entities that belong to a class. Nevertheless, catalogs can also contain meta data and an individual's context information. In the following table 6.1, we depict various selected catalogs that can be implemented based on our introduced ontology classes. More catalogs can be found in our PDD case study in section 7.5.

Class	Stakeholder	Key Concerns
`Abstract Method`	MGR, DE	Lists all recorded `Methods`. This includes the method's context, for instance, the connected quality attributes like the method's `Maturity` or `Repeatability`.
`Tool`	MGR, DE, KE, IT	Lists all recorded `Tools`. This catalog can be further subdivided into the precise class of `Tool`, like `Application Tool`, `Application Component` or a hardware / software distinction, of course.
`Abstract Resource`	MGR, DE	Lists all recorded `Resources` which can include context information as presented in the resource ontology on page 141.

Table 6.1.: Method Architecture Artifacts Catalogs.

Data Quality The following table 6.2 presents information that is either incomplete or unknown which helps the responsible KEs to identify lacunae in order to control the data quality. Data quality management using query languages like SPARQL and related languages like SPIN is a promising and effective approach (Fürber and Hepp 2010). Constraints are specified using SPARQL *ASK* or *CONSTRUCT* queries. Furthermore, the analyses help to detect entities that are either orphaned or unutilized, which is an indicator for managers and DEs in their strategical and operational planning.

Class	Stakeholder	Key Concerns
Orphaned `Abstract Method`	MGR, DE, KE	`Methods` that do not feature a `Concrete Method`, i.e., are not used in any `Process`.

Continues on next page.

Class	Stakeholder	Key Concerns
Orphaned `Abstract Resource`	MGR, DE	Lists all recorded `Resources` that do not feature a `Concrete Resource`.
Orphaned `Concrete Method`	MGR, DE, KE	Lists all recorded `Concrete Methods` that have missing links to either a `Process Action` or `Abstract Method`.
Orphaned `Concrete Resource`	MGR, DE	Lists all recorded `Concrete Resources` that have missing links to `Concrete Methods` or `Abstract Resources`.
Orphaned $\texttt{Tool}_1$	MGR, DE, KE	Lists all recorded `Tools` that are not related to any `Method`.
Unutilized $\texttt{Tool}_2$	MGR, DE, KE	Lists all recorded `Tools` that relate to an orphaned `Method`.
Missing `Method` descriptions	MGR, DE, KE	A list of `Methods` that do not feature a description.
Missing labels (names)	DE, KE	A list of `Methods` that do not feature a name (in all relevant languages).
Missing description of `Goal`	DE, KE	A list of methods that do not relate to a `Goal`.
Missing description of steps (`Procedure`)	DE, KE	A list of methods that do not relate to a `Procedure`.
Missing in- or output (`Resource`)	DE, KE	A list of methods that do not relate to `Resources` as input or output.

Table 6.2.: Method Architecture Artifacts Catalogs for Data Quality.

Matrices More powerful than catalogs of individuals belonging to the same class, are matrices that illuminate the links between the modeled entities. The following table 6.3 lists these relationships.

Class Relation	Stakeholder	Key Concerns
`Method` and `Business Service`	MGR, DE, KE	Lists the modeled relationships between `Methods` and `Business Services`.
`Method` and `Application Tool`	MGR, DE, IT	Lists all connections between `Tools` / `Application Components` and `Methods`.

Continues on next page.

Class Relation	Stakeholder	Key Concerns
Method and Technology Component	DE, KE, IT	Lists the infrastructure, where the Methods / Tools are deployed on.
Method and Business Resources	MGR, DE	Lists all Business Objects / Resources, indicated as a Method's in- or output.
Method and Process (Action)	MGR, DE, KE	Methods that are and can be used in a Process (Action). Because this query is based on the Concrete Methods, the related concrete quality attributes, e.g., the Process Suitability or Input Quality, can be displayed and made sortable for an assisted selection.
Concrete Resource and Process Action	MGR, DE, KE	Lists the modeled Resources and the Processes they are required in or produced by. Further information can link to the possible Concrete Methods.
Method and Goal	MGR	Lists all Methods and their influence on strategic Goals.
Method and Role	MGR	Lists all Methods and their related Roles, e.g., roles responsible, developers or operators.
Method and Actor	MGR, DE	Methods and their related Actors, e.g., people in the company that have already conducted the method, including an actors skill / experience.
Method and Organization Unit	MGR, DE	Lists all Methods that are either the responsibility of, or are used by an Organization Unit.

Table 6.3.: Method Architecture Artifacts Matrices of Class Relations.

Further artifacts The next table 6.4 features artifacts whose demonstration is most useful as diagrams or more complex UI elements, because they include many influence factors. As stated before, this compilation is not meant to be exhaustive and enterprises should implemented the introduced artifacts as seen required or develop new ones that accommodate their use cases.

Topic	Stakeholder	Key Concerns
`Method` and `Business Capability`	MGR, DE	Displays the `Method`'s influence on `Business Capabilities`.
`Method` ontology overview	MGR, KE, DE	Displays an ontology covering the methods.
`Method` chains and mixes	MGR, KE, DE	Displays possible method chains.

Table 6.4.: Method Architecture Artifacts Diagrams.

All the previously listed artifacts can of course be sorted, limited and filtered using SPARQL statements in order to display only the entities that are of interest to the current user. Furthermore, the listed artifacts are not exhaustive, i.e., they should be extended as required. Suggestions can be found in the related work about artifacts and views introduced in section 6.4.3.
Most of the relationships are self-explanatory, however, some need a bit more background information as elucidated in the following sections. For instance, the methods influence on the business capabilities has not been elucidated before.

Business capability A company's business capability is usually compounded by integrating different dimensions, i.e., people, processes and material/technology, as demonstrated in Figure 6.10. This capability's maturity can be measured

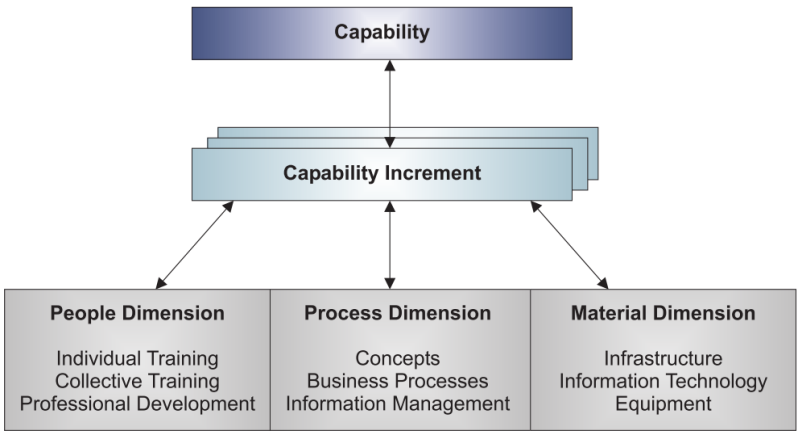


Figure 6.10.: Capabilities Increments and Dimensions (The Open Group 2011).

by various selected KPIs and is then compared to a capability maturity target during the TOGAF Architecture Development Method (ADM) (The Open Group 2011). As a matter of course, a company's ability to develop and apply design or

working methods is an influence factor on the company's entire business capability. The introduced method KPIs, for instance, method quality, maturity or other "ilities", can therefore be included into the formula of the process dimension for capability measurement, which is, according to TOGAF, enterprise-specific. Other factors are peoples' skills and experience when working with methods in the people dimension and the resources and tools required to conduct methods in the material dimension. Alternatively, a new method dimension can be introduced in contrast to the above mentioned option of introducing new method factors in the already established dimensions. One possible diagram, illuminating the current dimensional capabilities, is depicted in Figure 6.11.

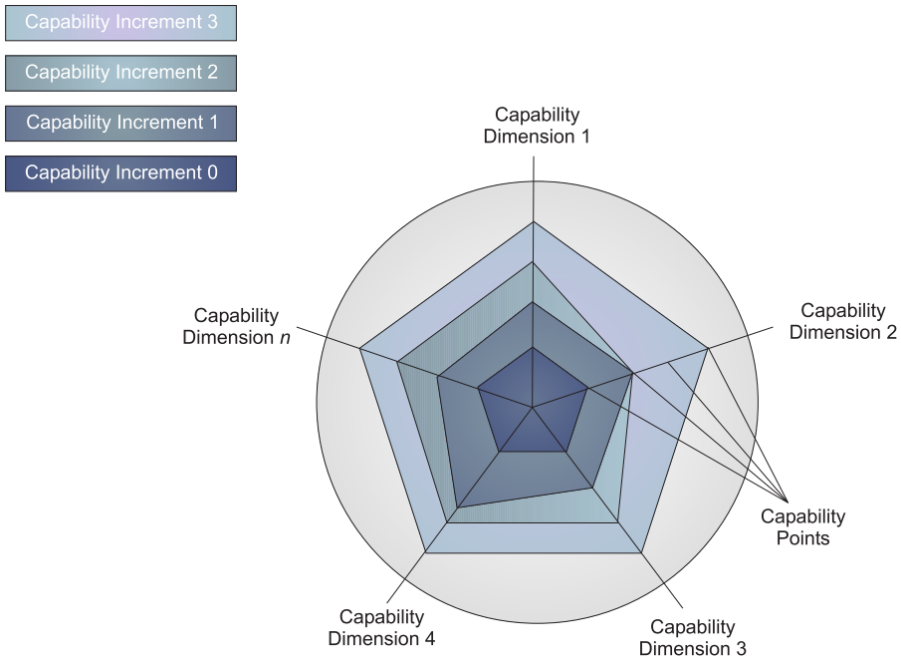


Figure 6.11.: Capability Increment "Radar" (The Open Group 2011).

Identifying and then coping with gaps detected in the baseline and target capabilities thereby influences method development. Furthermore, not only the capability of conducting or researching methods is an important factor, but also the knowledge about the enterprise's methods, i.e., developing and consuming the meta method architecture described in this thesis.

Method Goals The multiple advantages and possible supported business goals that can be pursued by combining methods and strategic business goals and objectives are elucidated in the subsequent section 6.4.4.

Ontology Overview The method ontology overview allows the user to discover the relationships on his own, by freely browsing the ontology, because not every use case is covered by artifacts. Frequently searched for relationships that are not yet covered by other artifacts should be identified and then provided as new artifacts, though. An ontology overview is depicted in our case study in section 7.5.4.

Actor and User Skills An important factor for the measurement of a company's method capability is the people dimension. By introducing a class `ActorMethodSkill` between `Actor` and `Method`, a people actor's skill and experience can be recorded and assessed. Next to the determination of the capability, this class supports, for instance, stakeholder management for projects. For a even more fine-grained analyses of the skills and experiences in the enterprise, a `User` class can be introduced.
Another use case of such an artifact is the retrieval of a contact person, a person responsible or a person in charge of a method.

Method Chains and Alliances One pivotal idea of our method framework is the combination of various methods in order to create method chains, and hence creating output from indirect input. In our case study, we have depicted how method chains can be visualized and how the logic can be implemented using rules and queries. Such a diagram is mainly valuable for DEs in a specific project, but also for senior business experts on a strategic level.

Methods in Processes An artifact that visualizes and lists the links between occurring and possible method usages in process actions supports the decision making of various stakeholders. This relationship is realized through the class `Concrete Method`. Hence, the related concrete quality attributes, e.g., method quality, duration or the process integration quality, can be displayed, as well.
A possible implementation of a visualization of methods that are used in a PDP is evaluated in the PDD case study in section 7.5.4. For instance, by considering the corresponding temporal or financial method attributes, the computed method chains aggregated values can help in process optimization, contingent on the depicted process action's cost or duration, respectively. This way, deviations can be detected and rectified, for example, if a method chain's output resources can be allocated earlier than formerly anticipated through the business process. Man-

agers also profit from this kind of artifact when performing strategical decisions, for example when promoting a virtual product development contrary to a physical development, because alternatives and the lack of them can be illustrated. Furthermore, such an artifact is an valuable support instrument for DEs, because alternative method applications on a project level can be suggested.

Methods and Business Objects By introducing methods into an EA meta model, completely new nexus of business objects/resources can be discovered. This can help to comprehend processes and hence to optimize them. Therefore, next to the matrix, introduced in table 6.3, a diagram depicting the relationships between business resources, that are solely based on methods inputs and outputs, i.e., method chains, can be of inestimable value.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT DISTINCT ?termURI ?prefLabel
  FROM <http://bioportal.bioontology.org/ontologies/NIF-RTH>
  FROM <http://bioportal.bioontology.org/ontologies/globals>
WHERE {
  ?termURI a owl:Class; skos:prefLabel ?prefLabel .
}

<http://NIF-RTH.owl/#nif_resourcep17:birnlex_2353> "Commercial ↙
    ↳ license"
<http://NIF-RTH.owl/#nif_resourcenif_resource:nlx_res_20090414> ↙
    ↳ "Biomaterial supply resource"
<http://NIF-RTH.owl/#nif_resourcep17:birnlex_2218> "3D ↙
    ↳ Time-series analysis software"
<http://NIF-RTH.owl/#nif_resourcenif_resource:nlx_res_090904> ↙
    ↳ "Reference atlas"
⋮
(omitted solutions)
```

Listing 6.1: Example of a SPARQL query for receiving preferred label properties (Salvadores et al. 2012).

Preferred Labels As mentioned earlier, all the queries and therefore the displayed information, is customized to the user. Not only by the above selected kind of method architecture artifacts, but also in the form of customized labels

for the concepts and individuals. An example for SPARQL query retrieving only the preferred labels is shown in listing 6.1. In our case study, we applied multilingual labels for all ontological entities.

6.4.4. Supported Business Goals

By making use of the TOGAF Motivation Extensions, companies can model strategic business goals, objectives and their drivers. The introduced class `Goal` of our method ontology can therefore be directly linked to either building block of a TOGAF `Goal` or `Objective`. While a goal is meant for strategic purposes and the mission of the enterprise, the objectives represent mid-term achievements that the enterprise wishes to attain (The Open Group 2011). Referring to the exemplary goals enumerated in the specification (ibid.), we itemize the goals that can be influenced by methods. For a real application of this enumeration, the appropriate goals should be selected, further goals appended, adapted and made SMART[2].

Improved Business Process Performance: By enhancing the re-use of existing methods, making their application-costs in processes transparent and predictable, having an assessable, consistent output quality and by increasing the process throughput through selecting the most appropriate methods for a task at hand. Besides, the separation of business logic and knowledge from code allows the right users to manage their own knowledge and thus react faster to business changes.

Decrease Costs: By reducing the redundancy and duplication of managed methods and by making them comparable. Consequently, cheaper methods, e.g, virtual replacements for physical methods, can be selected.

Improve Business Operations: A transparent method management makes the knowledge about the methods' durations, quality and maturity available which leads in turn to a shorter time-to-market, higher quality business information and better customer services.

Improve Management Efficacy: The modeled method knowledge leads to better and faster, substantiated decisions and offers more flexibility due to the available modeled method alliances, chains and alternatives.

[2]SMART is an acronym and procedure that is meant to make targets and objectives Specific, Measurable, Attainable, Realistic, and Time-bound and hence achievable.

Reduce Risk: By monitoring the methods quality and other "ilities", and decreasing errors in business processes due to the knowledge about method's inputs and outputs, risks can be decreased.

Improve Effectiveness of IT Organization:

- **Development**: The separation of code, business logic and knowledge heeds the different lifecycles of software and information which leads to less updates. In addition, methods themselves can be reused across the company instead of developing them separately or duplicating them.
- **Re-Use**: Managing a common SW-based KB can be re-used throughout multiple applications. Furthermore, a global method monitoring can lead to higher re-use of methods.
- **Resource sharing**: The KB can cover knowledge from multiple departments and divisions, which can be utilized by various stakeholders.

Improve User Productivity: Having access to shared knowledge about all available methods that can be conducted in a process action, including their conditions and attributes, e.g., required input and produced output, leads to a higher-quality method-usage and consequently can rise the user productivity.

Improve Interoperability: By following SW standards, other departments and domains can be easily mapped to the existing KBs. Furthermore, interoperability can be increased by providing a common infrastructure for the method applications, i.e., by managing and re-using the application and technology components where the methods are implemented on.

Reduce Lifecycle Costs: Code, business logic and knowledge separation allows an independent maintenance by the right people which leads to lower costs and faster reactions. Besides, orphaned or isolated methods and tools can be detected, interconnected with the remaining method landscape or replaced by a better alternative.

Improve Manageability: The modeled knowledge, analyses and artifacts enable managers or senior business experts a higher-quality strategic direction and planning and hence an improved development orientation towards an efficient and effective method use in the enterprise.

The above listed items show how an exemplary alignment of our method framework with an established EAM supports multiple business goals in a decisive way. As mentioned, this list is not exhaustive – other fields of improvement can be identified, for instance, in the disciplines of value engineering, cost–benefit analysis and other economic analyses.

6.5. Conclusion

Comparing and thus validating baseline and target architecture is a fundamental principle in EAM. Strategic planning of a method landscape in an enterprise, i.e., developing new methods, adapting existing methods, keeping efficient and effective methods or discarding useless ones, is also a fundamental course of action in the method management. Thus, expanding the steps explained in the ADM to our method ontologies, is a promising approach. For example, by performing a *gap analysis*, a technique widely used in ADM, EA architects can discover individuals that "have been deliberately omitted, accidentally left out, or are not yet defined" (The Open Group 2011).

Definition 6.3 (*Gap*)

"A statement of difference between two states. Used in the context of gap analysis*, where the difference between the Baseline and Target Architecture is identified." (ibid.)*

The prerequisites for this procedure can be conducted analogously to the ADM, e.g., by creating a target method architecture in addition to the existing baseline method architecture.
In the course of this chapter we have shown how the proposed method ontologies can be matched to an EA ontology that has been developed using the standards TOGAF and ArchiMate. Furthermore, expansions, like the motivation or product extension can be matched in order to profit from both KBs.
We have shown how methods can be selected, we introduced various method artifacts for all involved stakeholders, including artifacts that validate the model data, i.e., help to improve data quality. By realizing suitable queries for respective stakeholders, e.g., architects, KEs, method developers, IT experts or managers, views that show only the required concepts and entities can be created which lead to an improved and targeted display of information.
The EA concepts, for instance, user roles, actors, organization units, matched to our method ontology, enable users to find specialist contact persons, departments or knowledge carriers , for example, for the exchange of knowledge. The existence of a lot of appropriate methods for a specific task at hand does not automatically mean, that they are known or used for the process action. Usually, DEs just know a limited set of methods and tend to use only established ones (Albers,

Reiß, Bursac, Urbanec, et al. 2014).
Furthermore, methods are usually not integrated or incorporated in company goals and strategical decisions (ibid.). The supported and matching business goals and the methods influence on them makes it feasible to rise an enterprise's quality and maturity, e.g., by comparing the baseline and desired maturity and initiating a targeted method development that compensates or eradicates shortcomings and flaws.
We have developed and presented principles to perform method analyses in an EA, however, there are still a lot of areas that can benefit from this approach, such as an situation aware suggestion and selection of methods (ibid.), i.e., whilst taking into account the people with their experiences and skills while performing their tasks.

Part III.

Evaluation and Conclusions

"There are three principal means of acquiring knowledge [...] observation of nature, reflection, and experimentation. Observation collects facts; reflection combines them; experimentation verifies the result of that combination."

Denis Diderot (1713 – 1784)

7

Case Studies

7.1. Synopsis

For the creation of the case studies, i.e., the ontologies, entities, rules, queries and the prototypical applications, we applied the methodology introduced in chapter 5. Thereby, our methodology has been evaluated and improved. The major case studies in sections 7.3 and 7.5 each feature a subsection about their individual evaluation and outcomes.

The first case study presented in this chapter (section 7.2) deals with the Knowledge Management (KM) of ontologies, rules and documents, i.e., we showcase how an established traceability between source documents, e.g., policies explaining and stipulating a method application, the originating rules and a Domain ontology (DO) describing the applied concepts and entities support users in their daily work as explained in section 4.6, i.e., using ontologies as a Controlled Vocabulary (CV). Therefore, we rely on Semantic Web (SW) standards and illustrate how an ontology can be extended with linguistic knowledge.

In the major case study presented in section 7.3, we show how Semantic Web Technologies (SWTs) can be used to formalize legacy data and how to match different Computer Aided *x* (CAx) areas, i.e., Bill of Materials (BOMs) from the Computer Aided Engineering (CAE) and Computer Aided Testing (CAT) world.

Thereby, we evaluate the technological feasibility of the approach, used standards, tools and the methodology as mentioned. The integration of heterogeneous data has been necessary throughout the entire thesis and this case study showcases how this integration can be done on a data layer. Various methods in the field of CAx use a lot of different resources, e.g., data, as their input and produce resources as output. If these resources are integrated into our architecture, we can deduce and infer new knowledge based on this information, for example, these resources are managed in various CAx tools. Therefore, by conducting and implementing this case study, we demonstrate how matching former legacy data on one of the lower layers in our architecture can be achieved and thus leading to new knowledge about potentially unidentified relationships of elements on upper layers (*cf.* Figure 4.11).
In section 7.4, we present another brief case study. Here, we demonstrate, how existing established standard models, exemplified with Association for Standardisation of Automation and Measuring Systems (ASAM) Open Data Services (ODS), can be mapped to our method ontology which enables mutually beneficial analyses as described in sections 4.5 and 4.6.
The second and final major case study, the Property-Driven Development (PDD) use case in section 7.5, demonstrates how properties, product data, processes and of course methods can be matched and analyzed. Therefore, we implement a custom meta model that includes knowledge about methods and a specific kind of development process, the PDD. This case study involves most of the topics developed and introduced in this dissertation, i.e., a method meta model connected to a development process, the integration of present data, metrics that define the quality of method application, various views for different stakeholders and further concepts, like tools, data sources, documents, product information, processes and milestones which match parts of a company's Enterprise Architecture (EA). The principal matching of our method ontology and an EA, based on The Open Group Architecture Framework (TOGAF) and ArchiMate, has been shown particularly in section 6.2. Furthermore, by defining an appropriate set of rules and queries, we develop plenty of analyses in the PDD case study that cover many artifacts introduced in section 6.4.

7.2. Knowledge Management with a lexicalized ontology and rules

This case study describes a platform that helps industrial Domain Experts (DEs) to preserve the connection between textual sources and formalized Business rules (BRs) by using lexicalized ontologies both for links and for storage of the conceptual knowledge, i.e., it belongs mainly to the Knowledge Acquisition (KA) phase of the methodology. Furthermore, it elucidates the use of CVs as explained in section 4.6.
Generally, Business Rule Management Systems (BRMSs) are used to maintain and query business rules. In our approach, rules rely strongly on DOs, which model the business knowledge and provide a conceptual vocabulary for the formalization of the rules that are expressed in written policies. We show that lexicalized ontologies are a key component of such BRMSs and how such knowledge can be encoded.
Our proposed solution supports DEs in the automotive industry in understanding and maintaining their BRs by presenting the relevant passages of source documents that were used to create and define the ontological concepts. Besides, prepared texts, e.g., realized with (X)HTML and Resource Description Framework in Attributes (RDFa) (Herman et al. 2015), can make use of such an lexicalized ontology by highlighting and linking relevant terms to the conceptual model. The use case is based on a car development scenario, that resembles the PDD case study presented in section 7.5, which models the connection between car testing scenarios, e.g., safety tests, and the methods and tools used to analyze and prepare these tests.

7.2.1. Introduction

BRs can have different origins, such as regulations, policy documents or business logic directly entered by DEs. This business logic expresses both development processes and coherence between different events, including conditions and the resulting conclusions.
One of the main advantages of expressing the logic in BRs is that the domain knowledge is independent of the application code that uses this logic. In such

way, there is no need to alter the application code itself, when business logic evolves, new policies are applied, already introduced policies change or retire. Thus, the use of BRMSs leads to increased flexibility and agility of the organization.
However, DEs who are not also BRs experts may have difficulties expressing their knowledge in formalized logic languages. Supporting them in their management of the knowledge needed to write these rules is one of this approach's goals.
We propose building an ontology as a formal model for representing conceptual vocabulary that is used to express BRs in written policies. Such ontologies are shared conceptual models, so experts can share the same vocabulary. We use the Web Ontology Language (OWL) DL language to represent concepts and properties of the DO. In addition, the ontology is linked to a lexicon used to express rules in the text, so experts can query source documents. This calls for a formalism to link linguistic elements to conceptual ones. We opt to use the Simple Knowledge Organization System (SKOS) which provides basic elements to link domain concepts to terms from the text. The combination of OWL entities, SKOS concepts and their related information form a lexicalized ontology which supports the semantic annotation of documents.

Nota bene

This case study has already been published in Omrane, Nazarenko, Rosina, et al. (2011) and mainly reuses passages of this paper, either directly or indirectly. My contribution has been the provision of the use case, as well as the rules and linked documents, that have been processed by Université Paris 13 & CNRS. My role has been that of a DE at a car manufacturer that developed the conceptual model, i.e., the ontology, together with linguistic experts that acted as Knowledge Engineers (KEs). That means, the concepts of the various SW and Natural Language Processing (NLP) procedures conducted in this paper, i.e., the links between text and ontology, are the work of Paris 13. The results as well as the writing have been developed, discussed and improved by all involved parties. Conceptually, the use case has been a spadework for the major PDD case study presented in section 7.5. The mentioned paper also involves a *related work* section, which is not covered in this thesis.

Outline

In the following subsection 7.2.2 the business background and the motivation of this use case is described. The use of the standards OWL and SKOS in order to develop a lexicalized ontology is explained in section 7.2.3. Section 7.2.4 concludes this use case briefly and showcases an example of an annotated text.

7.2.2. Use case background

The motivation for this case study has been the same as described throughout this thesis, i.e., the challenge to cope with different methods from various CAx disciplines. This scenario is described more precisely in the PDD case study. Therefore, we point to section 7.5.2 for an exhaustive overview.
This case study deals with an arbitrary automotive domain, in this case, we decided to use regulations regarding the development and testing of seat belts and buckles from *ECE R16*. With the help of BRs, we calculate the duration of processes (*processDuration*) that involve different CAx methods (*cf.* Listing 7.1). Every CAx `Method` has an assigned attribute for either an estimated or an actual value for its lead time, cost and maturity (see also in section 4.4.3). Testing if the requirements for one vehicle or vehicle part (*here:* seat belt) property are fulfilled normally takes several process or method steps. As an example, the execution of a CAE simulation may take only minutes - the preparation, the modeling, and so on might take several weeks. Every process, e.g., an analysis or the validation of such an analysis, makes use of a `Method` (*cf.* Fig. 7.14 on page 241). The function *processDuration* calculates the whole process duration: the relevant attribute *time* is queried with the function *methodDuration* (which is defined in another rule) from all the methods involved in the relevant subprocesses.

```
processDuration(?X,?Y,?totalTime) :-
    methodDuration(?X,?Y,?totalTime).
processDuration(?X,?Z,?totalTime) :-
    methodDuration(?Y,?Z,?time) AND
    processDuration(?X,?Y,?previousTime) AND
    ?totalTime = ?time + ?previousTime.
```

Listing 7.1: Rule example, that calculates the total process duration.

With this rule, it is possible to visualize durations involving different methods that all test one specific vehicle (part) property. These indications support managers and DEs when planning their projects, so that they can choose whichever process and method is best suited for their work.
One of the difficulties with business knowledge rules is that various departments or roles sometimes use different vocabularies for the same concepts so they cannot understand each other immediately. Additionally, formalized rules per se are often not easy to understand. In this case, it might not be clear if *methodDuration* or *processDuration* also include lead and postprocessing time which is an information that can be acquired from the textual sources describing these methods.
Therefore, we use an ontology as a unifying model for a heterogeneous vocabulary and annotating the rules and hence reduce misunderstandings and ensure that people are discussing the same concept. Also, the users can easily confirm and verify the appropriateness of the modeled semantic relations. For this reason, this case study demonstrates how to handle links, i.e., establish a traceability, between source documents, such as policies, regulations and internal documents, and the concepts and instances of an ontology. Also, if a BR originates directly from a legal document, the relevant passage will be linked to the rule in the same way.

7.2.3. Expressing and linking the vocabulary

This section demonstrates how OWL and SKOS standards support formalization, document annotation, normalization and documentation that business experts face when designing an SWT-based application that also includes BRs. Therefore, ontological entities are extended with SKOS, for instance, for defining concepts or expressing alternative labels with terms that are extracted from source documents.
Obviously, the presented rules in Listings 7.1 and 7.3 use ObjectLogic, though. However, OWL ontologies are used to realize the conceptual model which can then either be transformed automatically into ObjectLogic or the rules and queries can be expressed with SPARQL which is exemplified in this case study, as well. During the case studies, we applied and evaluated several standards in order to express our business and domain logic resp. knowledge.

The diagram in Figure 7.1 gives an overview of the conceptual approach that links ontological entities with text passages of documents, such as written policies, by introducing a formalized lexicon in between.

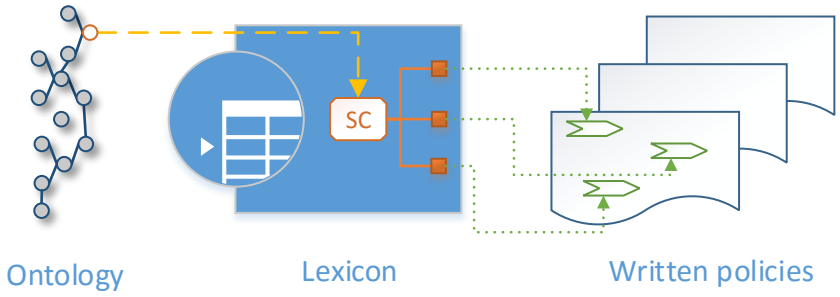


Figure 7.1.: A lexicalized ontology for annotating source documents. Each concept from the ontology is linked to a *SKOS concept* `SC` and each *SKOS concept* is related to its labels `l`. The annotations link some text entities to these labels.

Formalization of domain knowledge

An ontology not only provides the vocabulary to be used in expressing the rules, it also provides a structured vocabulary that encodes relationships between concepts and supports checking for inconsistencies. The following example in Listing 7.2 describes a concept `BuckleTest`, a method to test seat belt buckles, and its related SKOS concept with its preferred label "buckle test" in the text.

```
<owl:Class rdf:about="&onto;#BuckleTest">
  <rdfs:subClassOf rdf:resource="&onto;#PhysicalMethod"/>
</owl:Class>
<rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/ ↙
    ↳ terminae/Audi#BuckleTest">
  <skos:prefLabel>buckle test</skos:prefLabel>
  <rdf:type rdf:resource="http://www.w3.org/2004/02/ ↙
     ↳ skos/core#Concept"/>
</rdf:Description>
```

Listing 7.2: Formalized example concept `BuckleTest` and including SKOS extension.

Formalizing the BR vocabulary in an ontology gives a structure to and enables querying of the rule base. For example, we can display all the object properties including their domain concept involving physical methods by querying for an object property with the parent concept of all physical method concepts as range. Experts can also query the ontology itself to search, e.g., for a method that safeguards a property and is related to some constraints, as long as these properties are formally encoded. The following example in Listing 7.3 shows a query written in ObjectLogic. The concept `Product Information` is used in the safeguard `Process` and every `Process` uses a designated `Method`, i.e., a `Physical Method` or `Virtual Method` (*cf.* Figure 7.14). The result of this query displays all *ProductInformation*, i.e., car parts, functions, etc., and their related `Tools`.
Furthermore, the inference engine supports reasoning on the ontology. This enables searching for hidden information that is implicit in the rules, for inferring new knowledge, updating the rule base and ultimately improving the business of the organization. For example, experts may recognize that a safety test is less costly with some specific parameters.

```
@{Systemanalysis_ManagedProductInformation, ↙
    ↳ options[outorder(?Tool, ?ProductInformation), fillNull]}
?- ?Method:Method[utilizes_Tool->?Tool] AND ?Process:Process[
uses_ProductInformation->?ProductInformation, ↙
    ↳ uses_Method->?Method]
  AND ?ProductInformation:ProductInformation.
```

Listing 7.3: ObjectLogic query for receiving `Tools` and `Product Information`, related by a `Process` and `Method`.

Semantic annotation of documents

A key issue for experts in managing a rule base is to recognize that the meaning of formal rules and natural language sources, such as written policies and documentation is a precious source of information. It is also important to update the BRs as organizations often modify their policies according to internal or external constraints.
Therefore, it is important to be able to mine textual sources to understand how a given concept is used in business documents, what rules are related to it and how those concepts and rules evolve when the policies are updated. This is achieved

through the semantic annotation of the documents in which the mentions of the ontological entities (concepts, instances and properties) are highlighted and can be searched for.
Semantic annotation means that ontological entities are related to the terms that can be used to mention them in the texts and calls for designing lexicalized ontologies. When the ontology has been created from textual source, as for the use case ontology, it is easy to keep track of the terms that denote the various conceptual entities. The resulting lexicalized ontology is used to annotate source documents and to query them.
Our aim is to save the terms related to the conceptual vocabulary that is used to express the business rules. We do not need to encode sophisticated information such as the morphological structure of terms since we do not perform a deep analysis of the documents. We simply need to save the various linguistic units that denote a concept, instance or property.
This linking is achieved by using the SKOS standard as explained in section 4.6.3.

Normalization of vocabularies

When designing and updating BRs, experts face the problem of the heterogeneity of information sources and multilingualism.
Using SKOS allows expressing preferred, hidden and alternative labels and linguistic variants of the concepts as described in section 4.6.3 and exemplified in Listing 7.4.

```
<rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/ ↙
    ↳ terminae/Audi#SeatBelt">
  <skos:prefLabel>seat belt</skos:prefLabel>
  <skos:altLabel>belt</skos:altLabel>
  <rdf:type ↙
     ↳ rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
</rdf:Description>
```

Listing 7.4: SKOS preferred and alternative concept labels.

Defining concepts

Since experts often have to manage a large volume of information but do not always formally describe all the concepts, it is important to add informal documentation when it is available. Defining concepts in natural language is very important to understand what concepts mean, especially if they have ambiguous or implicit labels.

```
<rdf:Description rdf:about="http://lipn.univ-paris13.fr/RCLN/ ↙
    ↳ terminae/Audi#ReferenceZone">
  <rdf:type ↙
      ↳ rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
  <skos:definition>"Reference zone" means the space between two ↙
      ↳ vertical longitudinal planes , 400 mm apart and ↙
      ↳ symmetrical with respect planes , 400 mm apart and ↙
      ↳ symmetrical with respect to the H point, and defined by ↙
      ↳ rotation from vertical to horizontal of the head form ↙
      ↳ apparatus.</skos:definition>
  <skos:prefLabel>reference zone</skos:prefLabel>
</rdf:Description>
```

Listing 7.5: Concept definition by applying <skos:definition>.

Since legal documents, such as policies, often define their terminology precisely, we propose to extract those definitions from the source documents when designing the ontology and to associate them with the related SKOS concepts using the label <skos:definition>.
We followed this approach by analyzing and exploiting source documents, i.e., regulations, in order to find definitions for existing concepts in the use case ontology. For example, the concept ReferenceZone, extracted from the *ECE R16*, describes a "*reference zone*" as illustrated in Listing 7.5.

Developing the use case ontology

The ontology has been built in two steps. At first, the goal was to integrate the various existing knowledge sources into a single model. As a result, the Knowledge Base (KB) of our ontology consists of several thousands of instances sep-

arated into more than 30 concepts which describes a meaningful subset of the domain in order to perform the case studies.
In a second step, in order to better fit the experts' needs for semantic querying and document mining, the initial ontology has been restructured and lexicalized. Therefore, we have annotated a subset of these instances in English and German to allow the users to use the aimed-at application in their preferred language. Additionally, alternative labels can be added when different users prefer different terms in their daily work. It also appeared useful to increase the granularity of the domain model so as to represent not only the various types of tests but also their actual occurrences in the car manufacturing process (instances that are related to the different tests applied to specific vehicle models).
This led to encoding of various elements as concepts rather than instances (90 concepts were added). Modeling tests as concepts supports, for instance, querying of the ontology in such a way as to detect implicit relationships between tests, tools and parameters, which are important for the safety of vehicles. The conceptual structure has been reorganized (four subsumption levels instead of one). A SKOS resource has been associated with this resulting ontology: each concept is related to at least one preferred label and up to five alternative labels. In addition, using a subset of the initial ontology for the exploration of written policies showed that ten of the mentioned tests were missing in the initial ontology and led us to enrich it (Omrane, Nazarenko, and Szulman 2011).

Querying the ontology

Once the ontology is lexicalized, DEs can query source documents to search for fragments of texts that describe specific concepts mentioned in rules. For example, they can find all references of the concept `BreakingStrengthOfStrapTest` in the text, wherever it is mentioned in the documents as illustrated in Listing 7.6. The query returns all sentences that are annotated by that concept and use the preferred label "test of breaking strength".
Furthermore, DEs can search for all sentences where `Methods` are mentioned in the text. As the concepts expressing tests are subconcepts of `Method`, they can make use of this relation as exemplified in Listing 7.7.

```
prefix schema:<http://lipn.univ-paris13.fr/RCLN/terminae/schema#>
prefix onto:<http://lipn.univ-paris13.fr/RCLN/terminae/Audi#>
prefix skos:<http://www.w3.org/2004/02/skos/core#Concept>
select ?sentence
where
{  ?sentence rdf:type schema:sentence
   ?sentence schema:annotatedBy ?concept
   ?concept schema:realized concept <onto:BreakingStrengthOfStrapTest>
   ?skos skos:skosConcept ?concept
   ?skos skos:preflabel "test of breaking strength"
}
```

Listing 7.6: Example SPARQL Protocol And RDF Query Language (SPARQL) query for finding concepts that include a specific text.

```
prefix schema:<http://lipn.univ-paris13.fr/RCLN/terminae/schema#>
prefix onto:<http://lipn.univ-paris13.fr/RCLN/terminae/Audi#>
prefix skos:<http://www.w3.org/2004/02/skos/core#Concept>
select ?sentence
where
{  ?sentence rdf:type schema:sentence
   ?sentence schema:annotatedBy ?concept
   ?concept schema:realized concept <onto:Method>
   ?conceptfils rdfs:subClassOf ?concept
   ?skos skos:skosConcept ?conceptfils
}
```

Listing 7.7: Querying for subconcepts.

7.2.4. Conclusion

The use case showcased how users, such as DEs, can benefit from a lexicalized ontology that links to textual source documents, such as policies and regulation written in natural language. This way, ontological concepts can be defined precisely and alternative labels can be linked. Furthermore, the users can benefit from annotated text documents, because they have a direct link to the underlying conceptual model which supports understanding the domain.
Rules and queries that incorporate ontological entities which have been linked in such a way, are easier to understand and easier to write and maintain, because

DEs can check the underlying source documents for definitions, for example, the definition of a method duration in an ontology that is applied to test some seat belt which in turn may be linked to an internal policy.
For the technical realization of this approach we used OWL and SKOS to express the lexicalized ontology. Such an ontology can either be transformed into Object-logic, like the examples shown in this use case so that this language is used for rules and queries, or it can be queried with the SW Stack proposal SPARQL.
Thanks to the labels of concepts, the ontology can be used to annotate the documents. The technical realization of the annotation is not part of this use case, but can be achieved by using annotations in RDFa format in HTML as mentioned in Nazarenko et al. (2012). Furthermore, Nazarenko et al. (ibid.) contains a detailed methodology for applying further NLP techniques in order to acquire and develop documented BR that are traceable in textual sources. Figure 7.8 shows a text example extracted from the use case regulation *ECE R16* where all the mentions of known concepts are highlighted and linked to the respective ontological entities. This supports experts in browsing documents, because the conceptual domain model can clarify lots of relationships between arbitrary concepts as well as providing the vocabulary that is needed. With this technique, extracting and understanding BRs from textual sources is a much easier task for the involved DEs as well as the supporting KEs.

```
Two belts or restraint systems are required for the buckle
   inspection, the low-temperature buckle test, the
   low-temperature test described in paragraph 7.5.4 below where
   necessary, the buckle durability test, the belt corrosion test,
   the retractor operating tests, the dynamic test and the
   buckle-opening test after the dynamic test. One of these two
   samples shall be used for the inspection of the belt or
   restraint system.
```

Listing 7.8: A fragment of text annotated by the lexicalized ontology.

7.3. Semantic Integration of Bill of Materials

7.3.1. Introduction

During the Product Development Process (PDP) of a car many parts and assemblies are designed, tested and agreed upon. The ultimate goal for the final, ready for series production, parts is to write them in a structured list called engineering BOM which is handed over to the production division.
Many CAx technologies from sundry development teams are involved when developing a new car model and all those technologies, like simulation, technical drawings or real prototype design, will have to store their key results in the final BOM.
As expected, the individual worlds do not immediately match, since the CAx-specific BOMs vary in structure, granularity, vocabulary, semantics, up-to-dateness and completeness. This circumstance hampers our ability to share, interchange, compare or consolidate their contents. In fact, matching different BOMs, e.g., a Digital MockUp (DMU) and a CAT BOM, today is often a manual task and consumes lots of man power and time.
Each of those CAx technologies have their own viewpoint of a car and store this viewpoint in a CAx-specific BOM first.
Both, CAE and CAT, make use of Computer Aided Design (CAD) documents for the collocation of the specific BOM. Meanwhile, the CAD documents are continuously further developed. This means, that a snapshot of the documents at a given point of time is transferred into the other CAx world which impedes the traceability, because the source permanently evolves.
As shown in Figure 7.2 the various CAx-technologies use different tools that have their own representation of a car. One example is the AVx tool used in CAT for planning and preparing physical crash tests. Another example is the DMU system which is used to virtually assemble cars.
Usually, a good approach for combining, analyzing and comparing different data structures is to use domain-specific software solutions, e.g., Product Data Management (PDM) or Product Lifecycle Management (PLM) applications, connected to databases, that have their matching rules integrated. Different syntactical representations can be aligned and new matching rules can express the relations between various BOM's structures and entities. However, the Business Rules

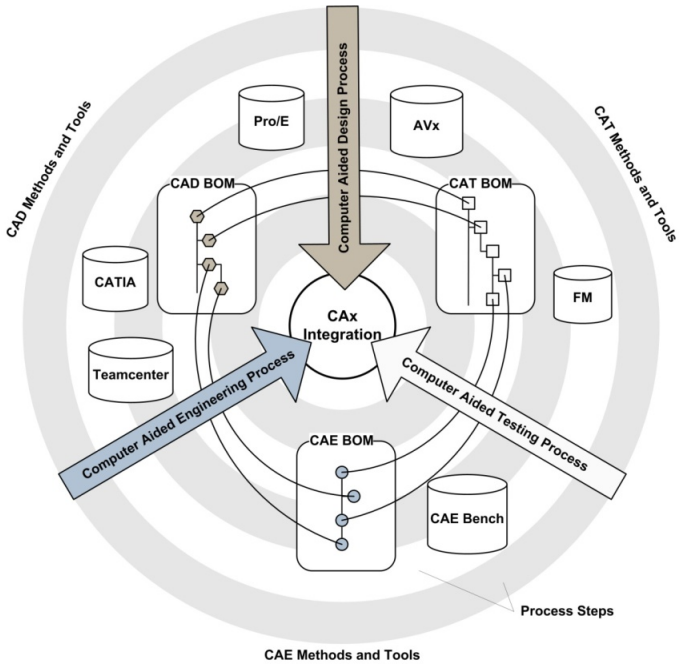


Figure 7.2.: CAx Methods and Tools (Rosina and Syldatke 2010).

Languages (BRLs) used in these applications are usually specific to the software application which leads to a "vendor lock-in" and thus a migration to another system is sometimes no easy task. Customized software solutions even have a lot of the BRs hard coded and thus cannot be adapted to new situations (and consequently new BRs) easily.

Nevertheless, a common approach, instead of using a PLM system, is to use spreadsheet software for managing and comparing BOMs. While using such a solution is often a fast success, it is error-prone, fragile, lacks a lot of features and hardly becomes comprehensible, understandable or sometimes even computable when the BOMs grow in size, complexity and quantity.

The goal of this case study is to showcase the feasibility of a SWTs-based approach for this task in a vehicle PDP.

The separation of the business knowledge in an independent KB and the application, which interacts with the user and presents the information, shall enable the right roles to maintain the different lifecycles. Besides, the showcased solution also demonstrates that product knowledge can be formalized with ontologies, queries and rules which is a requirement for combining this knowledge with other SWT-based solutions, like methods or EA ontologies. This can lead to even more precise and specialized analyses in more complex KBs in the future.

One of the requirements for the demonstrator was the functionality to import raw

BOM data, including AVx as well as DMU data. These files have been available in a Comma Separated Value (CSV) format, because they had been exported from an internal database for this demonstrator. This way, different exports for various car projects can be easily loaded into the demonstrator, thus mapped to each other, visualized and compared. This choice was made because the demonstrator should be loosely coupled to the existing Information Technology (IT) landscape. In section 7.5 another type of architecture is introduced which is more tightly coupled to the existing IT by connecting the databases with the ontologies via adapters. This way a strict separation of the terminological box (TBox) and assertional box (ABox) is possible and enables the maintenance of the data in its natural lifecycle. Certainly, it would be desirable to have a KB, like a triple store, as a primary source of information. This setup, however, is often not available.
Because complex product descriptions like BOMs tend to become very large, another requirement for the anticipated solution was the support for large amounts of data, i.e., a good scalability of the system.
To be able to estimate the success rate and coverage of the mapping rules compared to the overall entries in the BOMs, an overview of the statistics that represent the total and relative amount of the different kinds of mappings depicted in table 7.1 on page 222 has to be implemented. Thus, blank and weak spots in the mapping rules can be revealed and corrected.
Furthermore, the user should be able to browse through the different BOM structures and each node in the trees should be comparable one-by-one or one-by-many to its counterparts in the compared and contrasted with BOM. For analyzing the reason for the mappings, the fired rules should be displayed along with the source and target structure nodes. Besides, the various mapping rules should be selectable independently to display a filtered result list with the matching pairs.

Nota bene

This prototype and its evaluation have been realized together with *AUDI AG* during the *ONTORULE* project. Its outcomes have already been partially published in (Rosina and Syldatke 2010; Rosina and Syldatke 2011). The requirements elicitation, KA methodology, the modeling of rules and ontologies and the planning of the architecture have been mainly my own work. The implementation of the User Interface (UI) had been realized by a supplier.

Outline

In the introduction of this BOM case study, the reasons for developing a SWTs-based application and model that is able to integrate different BOMs have been described.
The following subsection 7.3.2 deals with the required foundational knowledge about generic and the particular BOMs used in this case study.
In the following subsection 7.3.3, the approach for solving the BOM-matching with ontologies and rules is explained.
Subsection 7.3.4 explains the architecture and technical background of the developed demonstrator. Additionally, the modeled example ontologies and rules are listed in chapter A of the Appendix. Furthermore, the created demonstrator is presented.
Afterwards, the application is evaluated in subsection 7.3.5 according to the specified business requirements, usability-focused objectives and ontology requirements in general. Finally, the conclusion summarizes this case study and its results.

7.3.2. Bill of Materials

An engineering Bill of Material (BOM) is a list of elements, i.e., parts and assemblies, which represents the content and configuration of a complex object, e.g., a car. The elements inside a BOM are typically segmented in (Eigner and Stelzer 2009c):

- *Core data:* data, that is autonomous, i.e., meaningful without a relation to other data, e.g., items or (generic and/or physical) parts. These parts include a technical description.
  At Original Equipment Manufacturers (OEMs)[1], like Audi, the core data consists of all atomic parts an object, e.g., the vehicle itself, is made of. Atomic parts are parts that are not disassembled at the OEM, but usually manufactured by its suppliers. The core data also includes all assemblies that emerge from the construction of atomic parts, as far as they are defined as independent parts.

[1]Commonly used for manufacturers that produce items that are sold by another company/brand. However, in the automotive sector the concept is used contradictorily, referring to companies that sell products which consist of parts purchased from third party vendors.

- *Structural data:* data, that relates the manifestations of the core data, e.g., BOM structures. This includes topics and annotations of the generic or physical parts that mainly serve for structuring. The structural data also comprises the information that represents which parts belong to which assembly.

The BOM structure describes the parts an object is made of and which parts are integrated in which object. Additionally, the location (place of assembly) and under which circumstances (variants, alternatives) they were integrated are declared. Each position is a so-called *usage*.
BOMs are used during all phases and disciplines of the PDP of a vehicle. However, the final development BOM is the structure where every agreed upon part and assembly of the to be build car model is recorded (*cf.* Figure 7.3). The final BOM is the engineering document that fully describes a car at the end of the PDP with all its parts and structures. This document is of high importance because it is handed over to the production business division and suppliers and is the foundation for producing the developed car in series.

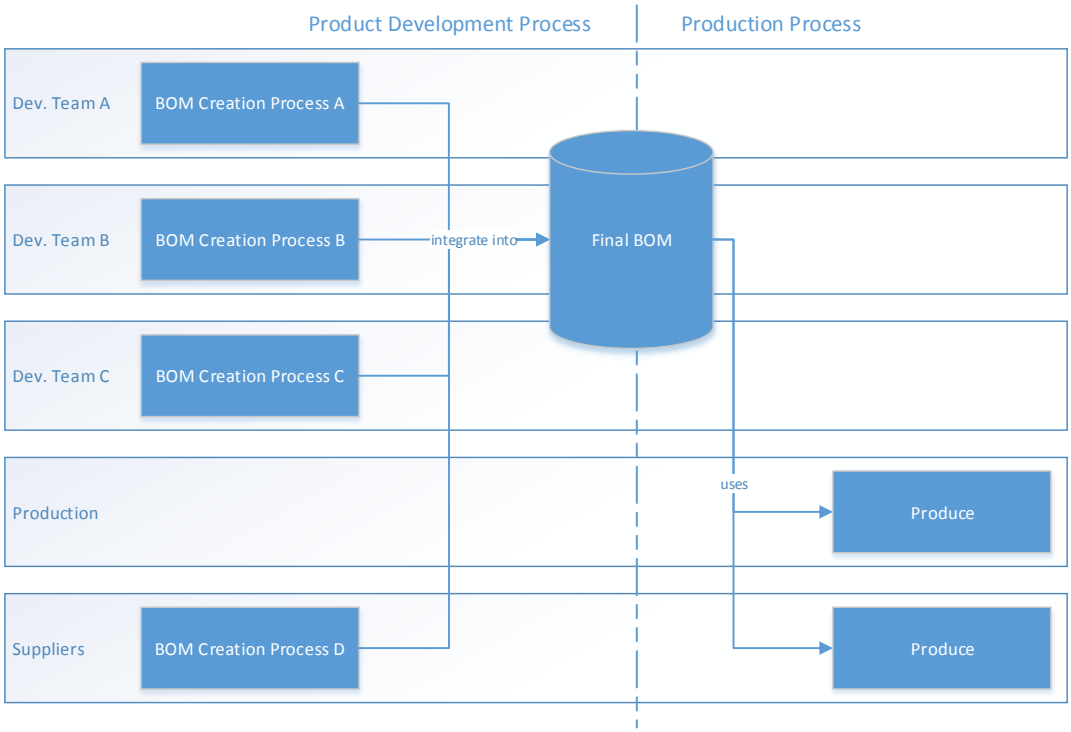


Figure 7.3.: Integration of BOMs.

Digital MockUp Bill of Material

Digital MockUp (DMU) is a method coming out of the aircraft industry to virtually assemble planes. In automotive companies this method is used to virtually assemble cars and in our case, it is also the name of a system, which provides this functionality. It consists of BOMs which are arranged in a tree-like structure.
A DMU BOM does not represent the parts required for a specific car but all parts for one car model. That means that all possibilities how this car could be assembled are included in this BOM, e.g., a steering wheel for right- and left-hand drive or a sunroof and non-sunroof version.
Additionally, a DMU BOM only consists of the parts that are relevant for the virtual construction (that means parts like washers, piping or bolts can be missing). In exchange, there is additional data like auxiliary lines that only appear in DMU BOMs. Furthermore, this BOM contains special parts that only appear at the beginning of the development.
All the DMU parts are discrete parts (there are no sets of identical parts), because every assembly location is destined for exactly one part.

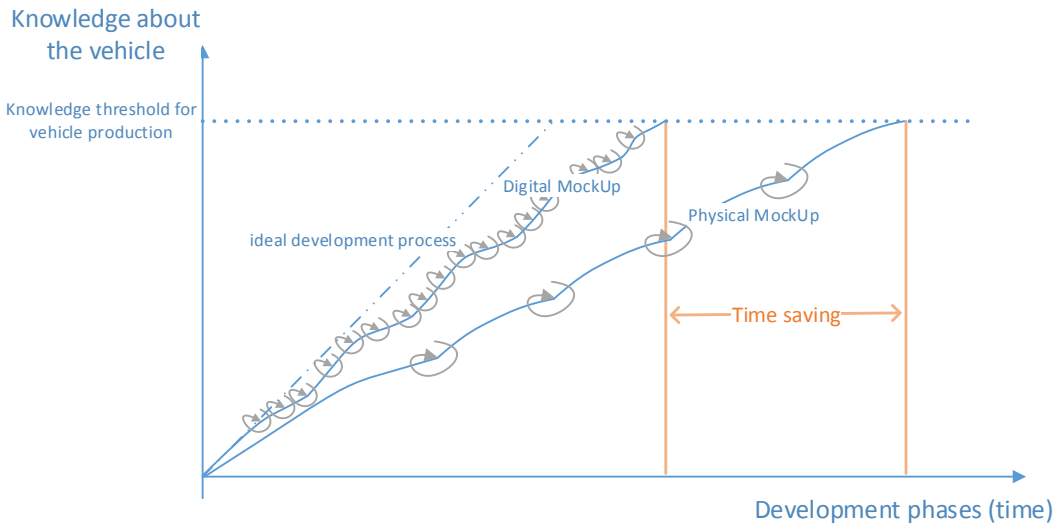


Figure 7.4.: Reduction of development time by using DMU. Based on Ovtcharova (2009).

Figure 7.4 illustrates how DMU speeds up the development, compared to a traditional physical mock up. Therefore, a strategic goal for enterprises that are designing complex products is to strive after less and less physical prototypes in the PDP.

AVx Bill of Material

AVx BOMs describe the assembly of a concrete physical vehicle and is used in an individual software. An AVx part is identified by its part number and the drawing date. The part number itself is not part of an AVx BOM, though, but is a concatenation of five different attributes.

7.3.3. Approach

A core function of the demonstrator is the visualization of the matching parts and assemblies, fed by a semantic KB in the background. To be able to continuously improve the rules, the demonstrator should also be able to analyze the exact and partial type of matches. One attempt to solve this matching problem is researched, tested and evaluated in this thesis in form of a demonstrator.
Ontologies are a great way to express the structure and vocabulary of a car within different viewpoints. For each of the viewpoints, an own ontology that reflects the specific BOM's structure is created (see Figure 7.5).

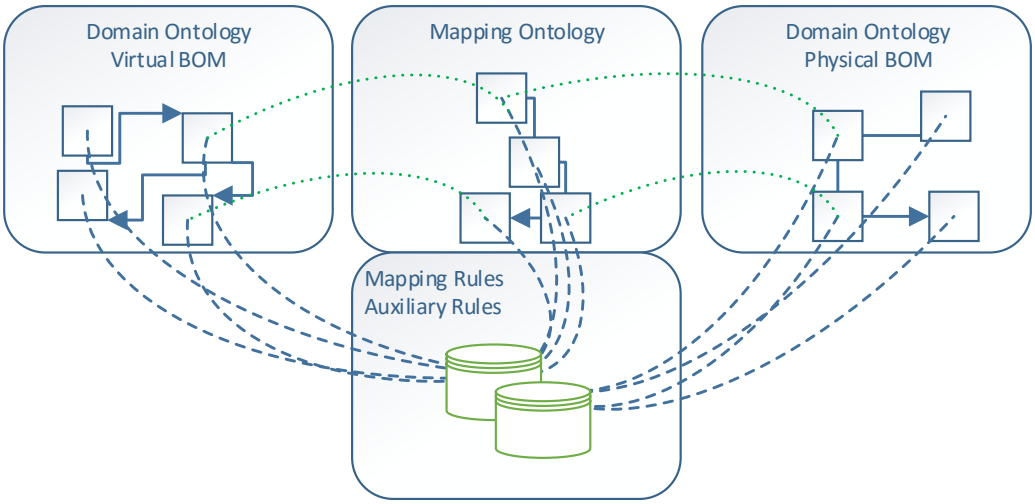


Figure 7.5.: Mapping virtual and physical BOMs with mapping ontology and rules.

A mapping ontology delivers a structure that combines these ontologies. For the actual mapping of the knowledge stored in the individual BOMs, rules are

created that express the mapping logic. These mapping rules are supported by auxiliary rules that filter, consolidate and reformat the individual BOMs' entries. The knowledge of this KB is retrieved by queries, that are also formulated in the same rule language, i.e., Frame Logic (F-Logic)
This knowledge model with its rules and queries shall be executed in a semantic knowledge and rule engine and visualized by a UI that depicts the individual BOMs and makes them explorable for a specific car model and project.

Knowledge Acquisition

Because the ontologies and rules required to solve this scenario cannot be conjured out of thin air, knowledge engineering methods had to be applied.
The KE's task was to build a correct and validated model that includes all concept definitions of the BOMs, the correct structure and the auxiliary rules and mapping rules along with the queries that allow comparing the different BOMs. Therefore, various sources have been considered and reused in this scenario.
First of all, textual documents that described the different BOMs were analyzed. These documents included detailed descriptions about the vocabulary used and the specific BOMs' structures. Another source consisted of database exports of the tools used for the different CAx technologies. For the sake of their confidentiality the data for the demonstrator has been distorted. Fortunately, also some existing ontologies that described BOMs and mapping could be reused and adopted.
After modeling a first version of the ontologies and rules with the help of the sources mentioned above, experts were interviewed to confirm and validate the mappings, concept definitions and BOM structures. Of course it would have been possible to model the ontologies and rules with the experts alone, but the KE's goal is to reduce their involvement and therefore their precious time to a minimum.
By applying the CogNIAM methodology (Nijssen et al. 2012) we validated the whole model. Experts were asked comprehension questions and had to confirm the correctness of each relation and class in the model by answering polar questions that covered all possibilities.
The KE gathered all the open issues and questions. Afterwards, the experts were asked various comprehension questions, i.e., "*Is each* DMU part *mapped to an* AVx part *or the other way around?*", "*How can a* DMU part *be mapped to more than one*

AVx part?", "*How can an* AVx part *be mapped to more than one* DMU part?" or "*Do you have a concept definition for the* Amount *of an* AVx part?".

Modeling and Design

During the KA phase, one of the major objectives was to gather and formalize the mapping rules between the different BOMs. By interviewing experts and analyzing documents and tools (*cf.* section 7.3.3) we were able to consolidate a set of mapping rules that allows us to compare the most important entries in the BOMs. The selected set showcases the feasibility of the approach and is an easily extendable foundation for future improvements.
First, an ontology that represents the contents and structure of the DMU BOM has been created. The facts and the structure have been generated with the help of DMU structure and model files which are not meant to be published.
Then, the AVx ontology has been produced that represents the structure of the already in place AVx database. The AVx system coordinates and documents the complete build and restructure process of objects that are defined in the BOMs. These are, for instance, cars, motors and gears. The AVx system is an individual software that supports the whole life cycle, starting with planning.
Afterwards, a mapping rule set has been developed that could handle the information of the above mentioned BOMs. The formalization in F-Logic has been done with the support of partners of the ONTORULE project.
The basis factor for comparing parts is their *part number*. Figure 7.6 depicts the structure of such a part number.

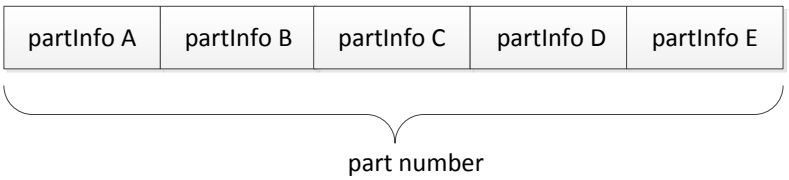


Figure 7.6.: Structure of the BOM part number.

The main difference between the part numbers in the AVx and DMU BOMs are their representation in the ontology resp. their representation in their source database. As depicted in Figure 7.7, the part number in the DMU BOM is stored

as a concatenated string that consists of five different part number information (A – E) that are stored separately in the AVx BOM. The base part number is a three- to nine-digit identifier, because the part number information B – D consist of a alphanumeric string with a varying length from one to three each. This base part number can be supplemented by additional information, i.e., an index which is represented by part number information E. This index stands for different part versions or describes specific codes used by the OEM. The part number information A is used to declare parts as a developer part and is empty otherwise. These parts are not relevant for the constructor and are therefore not matched to the other BOM's parts.

To be able to compare the two structures directly with the mapping rules, the auxiliary rule *setPartNumberAVx* has been introduced (*cf.* Listing A.3). The part number alone does not contain enough information to clearly identify a part in the DMU BOM. Additional properties, i.e., version, structural and source information, have been included in the ontology. The information is indicated by the properties with the abstract names like *property A*, *B* and *C* whose domains relate to the *Part* in the ontology extract of the DMU BOM in Figure 7.7. Another unique characteristic of a DMU part is its drawing date, for instance, the release date or an archive date. But even with the distinction of the part number and these additional properties, it is still possible to encounter duplicates in the DMU BOM. This is due to the fact that in some cases more than one instance of a specific part is needed for the virtual construction and the DMU BOM does not comprise indications of quantities.

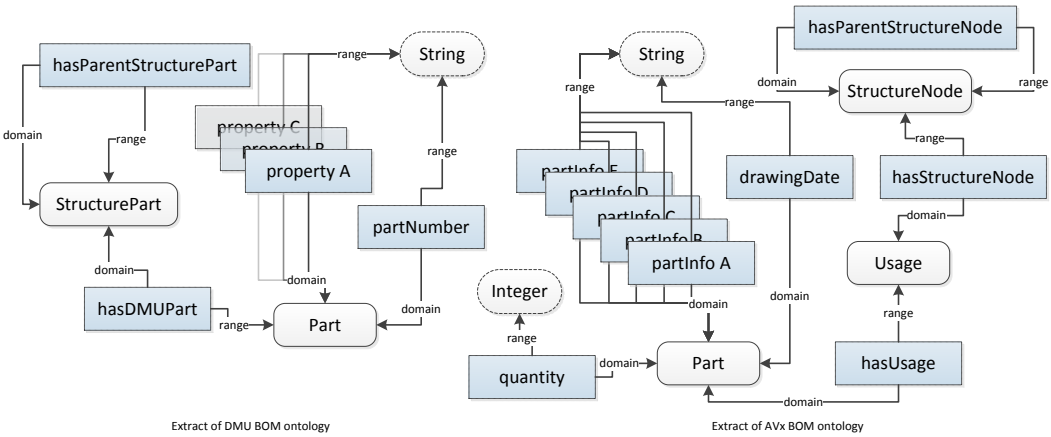


Figure 7.7.: Extracts of DMU and AVx BOM ontologies.

Due to the modeled auxiliary rules that instantiate the part numbers in both BOMs, we are finally able to compare them according to the mapping rules.
All the parts of the DMU and AVx BOMs shall be combined using the mapping rules in table 7.1. The table depicts the possible types of mappings for the three categories of matches: *mirrored part*, *exact* and *no index* matches if there is a match. Otherwise, if there is no counterpart for a DMU or AVx part in the other BOM or the part is ignored, one of the *no match* categories is used.

Category			No Match	Match		
				Mirror	Exact	No Index
Left Part				x		
Right Part				x		
1 *DMU*	$N:1$				x	x
	$1:1$	$q \equiv 1$			x	x
		$q \not\equiv 1$			x	x
$N:M$	*valid q*				x	x
	invalid q				x	x
DMU Developer Part			x			
DMU Part			x			
AVx Standard Part			x			
AVx Developer Part			x			
AVx Part			x			

Table 7.1.: Mapping Categories.

The preferred type of match is a *mirror* match (see the implementation in section 7.3.4). That means we have two AVx and two DMU parts that represent mirrored parts of a car. One is located on the left side, the other one on the right side. Parts that are found by *mirror* rules are not considered for the exact matches or any other category anymore. A *mirror* match has a higher priority than an exact match, because the other way around one matching pair would either belong to two different matching categories (*mirror* and *exact*) or we would not find mirror parts at all (they would all be *exact* matches).
During the KA phase, the rule that identifies mirrored parts has been formulated as illustrated in Example 7.1.

Example 7.1 (*Mirror parts rule in natural language*)

"AVx parts are mirrored parts if one of them is assembled at the right, the other one on the left side of a vehicle. In an AVx BOM, this circumstance is expressed by the attribute partInfoC*. If the whole part number is the same except for the* partInfoC *and the difference of these two* partInfoC *entries is exactly one (the right part has a* partInfoC *increased by* 1*) then the part with an odd* partInfoC *is the left mirror. The part number with a* partInfoC *increased by one is the right mirror part."*
Example: *AVx left mirror part:* 4*L*3991105 *and AVx right mirror part:* 4*L*3991106*.*

This natural language based rule example has been formalized in order to identify such entries in the following implementation section 7.3.4, more precisely Listings A.1 and A.2.
The second best type of match is an *exact* match which means that the exact same part number is found in the other BOM.
If that is not the case for a specific part the third type of match is the *no index* match which means that we do not consider the index component of the part number while comparing two parts of the two different BOMs. The index represents a version of the part. If it is different and used for the assembly this means that this part is still compatible to the rest of the connected parts. Normally only minor changes are being made when the index changes.
Supplementary, these three different mapping categories are refined into subcategories for even more precise distinction.
The subcategories like $N:1$, $N:M$ and so on represent the quantity of mappings that have been found. So it is possible that one DMU part maps to several (more than one) other AVx parts ($N:1$) or one AVx part maps to several DMU parts ($N:M$ *invalid* q). If there are several equal parts needed for the physical assembly then they are not listed x times like in the DMU BOM but use an attribute *quantity* to express the number of parts required. If the amount of matches is the same as the *quantity* the category $1:1$ $q \equiv 1$ (respectively $N:M$ *valid* q) is used. If there are more or less matches then that type of match belongs to the $q \neq 1$ resp. *invalid* q category.
As in the DMU BOM, there are parts in the AVx BOM that have no counterpart in other BOMs so the mapping rules only apply to a subset of all the entries. This is because either the DMU parts are developer parts and not meant for the real assembly or the parts are standard and norm parts from component suppliers.

These are parts that have no counterpart in the DMU BOM, like piping, bolts and other small components. Additionally, normal AVx parts do not always have an equivalence due to the fact that mapping information is still missing or has not been formalized yet. Another possibility is that the quality of the CAD (DMU) drawing is not sufficient or the drawing itself not even possible.

7.3.4. Implementation

The goal was to implement a prototype application[2] that provides a visualization for mapping relations between different BOMs, based on the semantic knowledge in the background. After the elicitation of the business knowledge, the ontologies for the different CAx technologies, the mapping ontology and the various rules and queries had to be modeled (*cf.* Figure 7.5). Further requirements next to the business objective concern the UI and the IT architecture including the format and platform choice. We realized the project with the semantic middleware OntoBroker, which came along with the ontology and Logic Programming (LP) language F-Logic (*cf.* section 2.3.3).

Architecture

The separation of code and business knowledge led to a three-tier architecture (*cf.* Figure 7.8) with the web-based application including the UI in the upper tier and an intermediate service tier that connects the application and the KB in tier three. In order to be as autonomous as possible, the different tiers communicate via web services through sundry interfaces. Indeed, the web-based application itself comprises even more architectural layers since it has been developed using the Model-view-presenter (MVP) paradigm, but the applied software design pattern is negligible for this case study.

[2]The demonstrator can be tested online at http://ontorule-project.eu/caxdem/; last accessed 10/29/2014

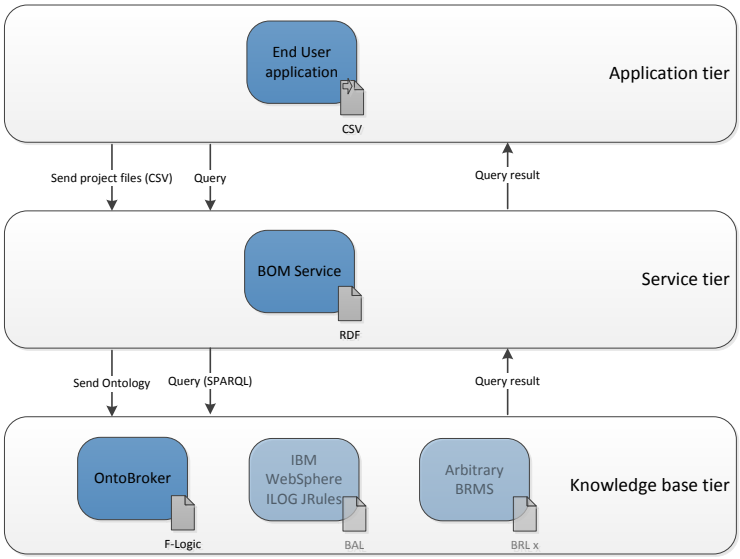


Figure 7.8.: BOM demonstrator 3-Tier architecture.

The prototype application has been realized with Java and Google Web Toolkit (GWT)[3]. This enables an easy, platform independent and web-based interface for analyzing mappings between BOMs. To improve visualization of data we additionally used the GWT extension GWT-Ext. The application can be used to compare any vehicle project represented in different BOMs. The data can be imported using a wizard that accepts CSV files. These files are then forwarded to the Service Tier where the TBoxes of the to be filled BOMs are already stored in Resource Description Framework (RDF)/OWL. The BOM Service transforms the imported data into the corresponding ontologies' ABox. By introducing the Service Tier, the application is more loosely coupled from the underlying KB tier which allows a straightforward substitution of the connected semantic middleware or BRMS. In this demonstrator's architecture the back-end of the end user application and the BOM Service both run on a Tomcat server[4].
In the implemented scenario we decided to use OntoBroker 5.x by ontoprise GmbH as an example SW middleware together with the knowledge representation language F-Logic in the KB tier. Alternatively, we could have used IBM WebSphere ILOG JRules and it's BRL Business Action Language (BAL), as IBM ILOG has also been our ONTORULE partner, or any other arbitrary BRMS, like Apache Jena, that is capable of importing RDF/OWL ontologies and offers a BRL that interacts with them. The connection from the BOM Service to the BRMS was

[3]See http://www.gwtproject.org/ for details; last accessed 10/11/2014.
[4]Apache Tomcat: http://tomcat.apache.org/

established by operating OntoBroker as a web service. The mapping rule set has been directly loaded in OntoBroker on startup. That means after selecting a vehicle project in the UI, its data is transformed into an RDF/OWL ontology and this ontology is send to OntoBroker where it is processed together with the formerly designed rules.
The downside of this approach is, that the mapping rules would have to be rewritten in another BRL when the BRMS is substituted. Unfortunately, Rule Interchange Format (RIF) has not been matured enough to write the rules application-independent, resp. interchanging all rules between the applications.
After this initialization the connection works as follows: After each request for mapping information that is triggered by the user the BOM Service sends a SPARQL query to OntoBroker which in return sends the result and an explanation consisting of the rules used which is then visualized in the UI.
In total there are three different services: The raw BOM data has only been available as CSV files (database export) which means one of our first tasks was to create ontologies that represent this data so we could take advantage of the many benefits.
After the structure has been analyzed and the ontology TBox has been designed we were able to develop an automatic translator from our data format into an F-Logic ontology which is part of the BOM service. The target language can be easily exchanged so it would be possible to export an RDF/OWL ontology, too. This way the application layer is decoupled from the rule engine and BRL used. For the sake of simplicity of the demonstrator we only developed a connection to OntoBroker from ontoprise GmbH, though.
The raw BOM files can be imported with the provided UI and are then automatically transformed and forwarded to OntoBroker.

Projects & Import The prototype application data context is called project. Projects are domains in whose context a mapping analysis is performed. Thus, each car model implies a separate project.
Projects are initialized with a set of raw data (BOMs). Therefore, the UI provides import functionality for BOM files. Uploaded files are sent to the mapping service interface. The AVx and DMU BOMs are represented by different CSV-files.
To simplify upload interaction, the UI provides functionality to upload archives containing all the BOM files at once. The required file for building the structure trees are resolved automatically by evaluating the file name.

Structure trees The main user interaction is done by using the structure trees. The AVx and DMU trees are initialized with the nodes of the structure related to the current project's BOM. Clicking on a tree node executes the following actions:

- If the node has child nodes, the child nodes are displayed as the leaves of the selected tree node.
- Selecting a structure node (AVx as well as DMU) starts a mapping query with the selected node as the mapping source.

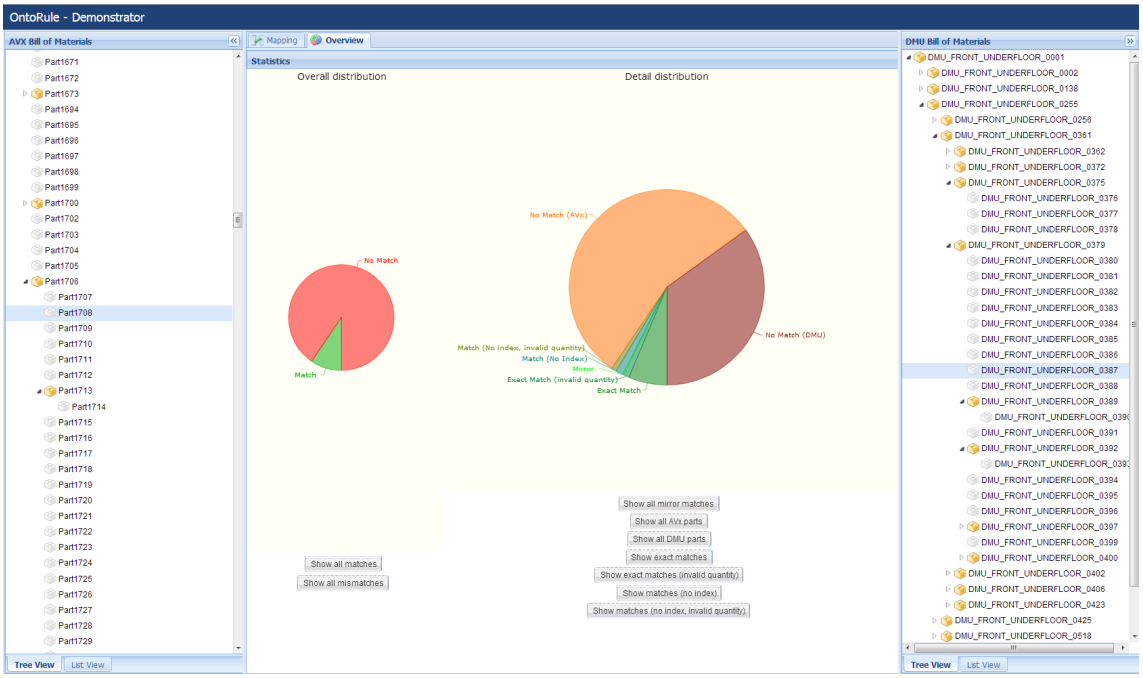


Figure 7.9.: BOM Matcher showing statistical analysis result in pie charts.

Mapping The mapping is done in an external service that is accessed via an appropriate interface which is depicted in Figure 7.9. The service provides the list of mappings related to an input query as well as statistical information like the total number of (mis-)matches for a specific project, visualized in the depicted diagrams.

Each query result returned by this service refers to a list of resolved mappings and includes information about the source node which is the selected AVx or DMU nodes, the target node which are mappings to DMU or AVx nodes and a textual description that provides information about the reason why this concrete match or mismatch was found.

The UI provides functionality to filter the displayed mappings by the different (mis-)match types, e.g., no index, introduced in table 7.1.

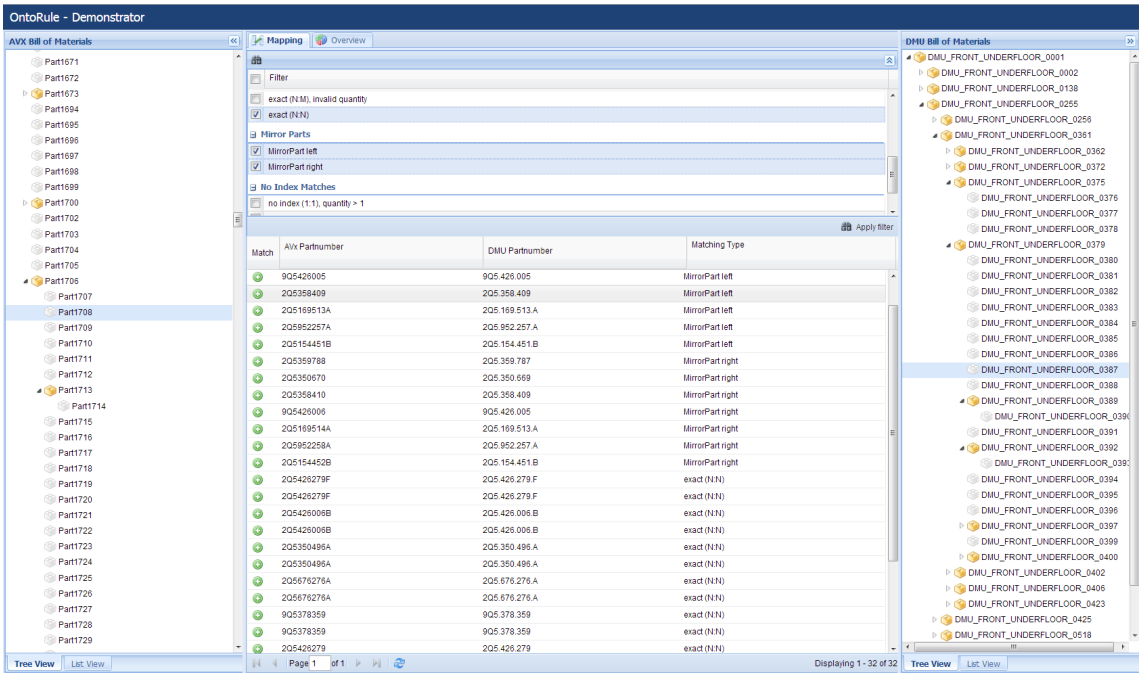


Figure 7.10.: BOM Matcher showing different structure trees and mapping rules.

Implemented Rules

In order to calculate and infer the various relationships between entities, we do not construct complex queries, but materialize the relations beforehand with selected rules. This way, we can enrich our KB with the required information for displaying all matches and reduce querying time for the real time retrieval. Therefore, the ontological classes are expanded by newly introduced object and data properties in the TBox. The actual relations in the ABox are inferred by the rules.

All the hereafter introduced example rules are located in the annex in section A.2 for a better readability.

For instance, one part of the AVx BOM shall be identified as a mirror part of another part in the AVx BOM via the rule "partNumberMirroredPartAVx" (*cf.* Listing A.1), i.e., two parts in a vehicle project that are defined as the exact same part, just mirrored (see Example 7.1). This rule alone, however, does not include enough information to clearly identify two parts as "mirror parts". Another rule

has to be considered (*cf.* Listing A.2) which adds additional constraints by testing whether the part is a developer or standard part. These are types of parts that are not allowed to be declared as mirror parts and are implemented in another rule. These two rules add new information to existing parts when the conditions hold true, i.e., they add the relations *#partNumberMirroredPart* and *#mirroredPartAVx* to all the individuals in *avx#Part*.
Another example, that has been mentioned in the previous section 7.3.3 and in Figure 4.15 of chapter 4, is the property `partNumber`. The rule depicted in Listing A.3 adds a new property `partNumber` which consists of the five individual part number information to every AVx part. Because the part number information *E* may contain white spaces, the rule trims *E*, i.e., deletes these white space occurrences, before concatenating the whole string.
Besides these examples, we implemented all required rules that are needed to query the categories listed in table 7.1 and created properties that resemble their relations for each entity. This way, the queries needed to display the UI information could be kept simple, because all relevant information was directly attached to each entity.

7.3.5. Evaluation

The obvious objectives of this case study were the successful formalization of individual BOMs using SWTs and the application of the formalized business logic to achieve the mapping and therefore integration of different BOM structures.
Being able to combine these different domains using SWTs is the foundation for combining CAx disciplines in general. Thereby, we can detect relationships that are potentiality unidentified on a higher, more abstract operational or strategical level and vice versa, if the business link is known, we can justify and prove those connections.
Furthermore, this case study has been used to review, evaluate and optimize our introduced methodology.
Besides the business requirements of the case studies, we had defined several ontological requirements, that can be separated into the three mentioned evaluations aspects in the following enumeration (Sure, Staab, and Rudi Studer 2009).

- *Technology-focused Evaluation:* This type of evaluation focuses on ontology properties, e.g., the correctness of the syntax of the ontology or the consistence of the ontology, i.e., the semantics. Another aspect of this type of evaluation focuses on the application of the ontology, e.g., its performance, scalability, interoperability etc.
- *User-focused Evaluation:* Evaluates the satisfaction of the users when working with the ontology (application), i.e., the usability of the ontology-based application compared to a classical IT application covering a similar functionality.
- *Ontology-focused Evaluation:* This sort of evaluation focuses on more philosophical aspects of the ontology design, validating the taxonomic relations and class-instance relations. One such methodology is *OntoClean*, that validates the ontological adequacy by considering philosophical entities, "like essence, identity, and unity, which are used to elicit and characterize the intended meaning of properties, classes, and relations making up an ontology" (Guarino and Welty 2009).

When all three aspects of the evaluation have been performed, the ontology is completely evaluated.
As described, we successfully integrated different BOMs of different Research & Development (R&D) domains. These BOMs were represented in different structures — a DMU BOM is separated into the car's physical zones, like the greenhouse, cockpit or underfloor. An AVx BOM is a BOM that is used for assembling and constructing real prototypes in CAT. Therefore, the order of the parts is orientated towards the actual assembly of the car, meaning the parts that have to be used first in the creation of the model are also first in the BOM. The F-Logic rules that we created helped to map the same parts from different BOM structures, vocabularies and granularities and even allowed us to map similar, but not identical, parts by defining complex comparison criteria. DEs were able to comprehend complex matching scenarios due to a visual display of the rules that fired and hence which BRs were responsible for the specific mapping.
Another measurement of success that was specified was the question if the solution for integrating BOMs could be reused for CAx integration in general which can be answered positively. This requirement arised because, in addition to BOMs, a car manufacturer uses a lot of other product information in various systems to describe the PDP.

As this SWT-based application was only the first step into creating a larger and more complex business application, i.e., integrating methods along a PDP, we mainly focused on evaluating technological aspects to determine the fitting platform and format choice.
One such requirement was the scalability of the system. Comparing different BOMs can easily involve several 10,000 parts along with their meta-information and structures. With the developed demonstrator (Rosina and Syldatke 2010) we successfully integrated representative BOMs of different domains.
As described, the consistency and integrity could be verified using appropriate tools, i.e., OntoBroker and OntoStudio, and by creating particular queries for measuring the data quality, i.e., for discovering the mapping coverage and orphaned or otherwise unmappable individuals.
The methodological approach has been proven useful and successful as we were able to acquire and model the relevant knowledge. DEs have been interviewed, knowledge extracted from existing database dumps, ontologies and documents. The modeled ontologies and rules, based on these sources, could be validated by further DEs and verified using the mentioned tools. For this purpose, we modeled rules and queries that examined and tested our data quality and integrity. Furthermore, when new mapping knowledge became available, we were able to extend our rule set without interfering with the domain knowledge and IT application.
The execution of the ontologies and rules has been successful, as well. A benefit for the stakeholders, in contrast to conventional applications, has been created by separating knowledge, UI and utilizing a service-oriented architecture, all the tiers can be maintained and exchanged separately by the appropriate roles and by considering their individual lifecycles.
With the next demonstrator (*cf.* section 7.5) we combined product information, like functions, concept sizes and most importantly CAx methods in an ontology and successfully used the integrated knowledge to safeguard the achieving of a vehicle's target properties in the PDP. Furthermore, we re-apply the methodology to perform KA, modeling and execution.

7.3.6. Conclusion

Figure 7.11 illustrates the thesis' elements which have been applied and evaluated in this chapter.

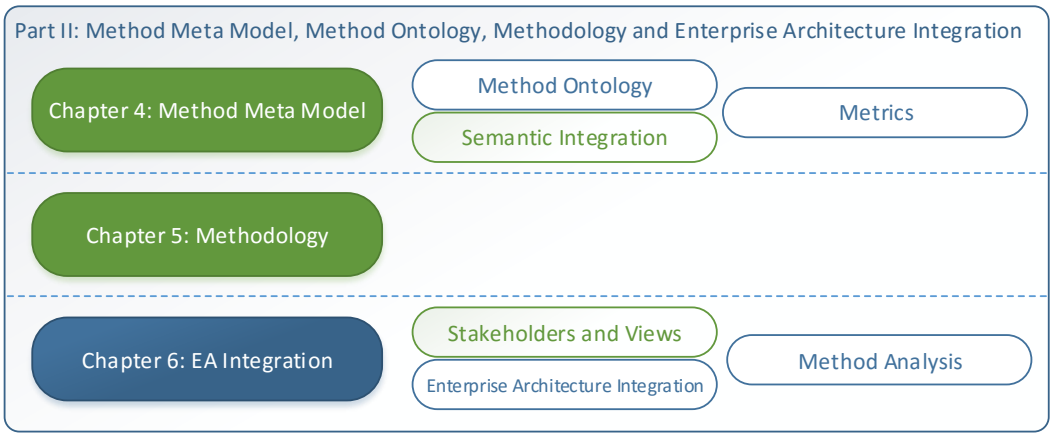


Figure 7.11.: Evaluated thesis structure elements for the BOM case study. Topics highlighted in green are applied here.

The usage of ontologies and rules to express BOMs and a methodological approach to design, model and apply them has been evaluated in this case study.
Two different BOMs, the AVx BOM of the CAT background and the DMU BOM of the CAD background, have been modeled. This has been done in order to develop an application that is independent of the vehicle we want to compare and the type of CAx-technology involved. Being able to load different rule sets into the connected rule engine allows us to compare not only AVx and DMU BOMs but also different systems and CAx-technologies. It also allows us to modify the rules when new knowledge becomes available without changing the developed application itself.
The use of ontologies which act as the data source for a specific vehicle project makes it possible to easily compare different types of vehicles.
With our approach, we developed a demonstrator that is independent of the BOMs used and the mapping rules needed to combine them. The BOM data about a specific vehicle is stored in an ontology which makes it easy to substitute the type of car we want to compare. The rule sets with the mapping information for comparing the sundry CAx-technologies can also be substituted which enables us to use the same application to analyze connections between different systems in the various CAx technologies, e.g., CAD–CAT or CAE–CAD.
Furthermore, the BOMs, mapping rules and method knowledge are never final. When new knowledge becomes available, existing queries can be adapted to the emerging situation due to the separation of application code, rules, queries, and

the KB in form of ontologies. The primary goal is always to have an exhaustive match between DMU and AVx BOMs. It is just hard to achieve, because the virtual drawing world and the physical car part world are two different things, with mostly asynchronous life cycles.
This prototype demonstrates the feasibility to identify (missing) mapping information between various BOMs and Computer Aided (CA)-technologies using ontologies and rules in order to increase the flexibility, quality, and efficiency of the PDP. Furthermore, it supports DEs by visualizing the results of the mapping rules which enables them to validate their semantic correctness.
The knowledge can be maintained at a semantic and autonomous level which is a benefit compared to the syntactical representation used in classical applications because this knowledge can be reused in other projects and thus helps to reduce cost, time and effort.
The transformation of data, structure and knowledge towards SWTs, like ontologies and rules, bears many advantages in general. DEs in R&D can manage, change and add the vocabulary and business logic and re-adjust it to the dynamic business parameters on their own, separated from the IT application, when it becomes available. Th application just acts as a layer to present the knowledge, considering performance, visibility and security issues, and to interact with the user. The transformation into standardized formats, like OWL, SPARQL or in this case F-Logic, allows reusing, sharing and interconnecting the knowledge across sundry domains and projects. Complex and multidimensional coherences and mapping rules can be expressed with SW languages and technologies, which is hard to implement and maintain, for instance, in classical spread sheet software – a still common approach, nowadays. Besides, the familiar characteristics, like correctness checking, reasoning, structural adaptability, meta information, annotations and the extensibility, are advantages of this approach. Furthermore, the IT tools themselves can be reused with additional and different knowledge, because business logic and data are not hard-coded into the application. This loose coupling approach allows the KEs to test the behavior of newly implemented rules and queries beforehand, independently of the application, supported by query and rule explanations of the BRMS.
Being able to adjust mapping rules of different BOMs from various CAx domains leads to a higher flexibility of the overall PDP and a better integration of virtual development. The developed demonstrator helps to identify the matching success rate and thus enables the DEs and KEs to further improve the KB.

However, modeling the BOMs with ontologies and rules is just a first step into integrating the enterprise data and knowledge in a SWT-based way. This includes monitoring and planning systems, like Enterprise Resource Planning (ERP), EA and PLM systems.

7.4. Aligning the Method Model with Industrial Standards

In order to demonstrate the reusability, relevance and adequacy of our method ontology, we compare and map the model with well known and de facto standards in the industry. These standards describe what we understand as methods partially, sometimes with another point of view, but it is often possible to align the core of their method semantics to ours.
The matching or incorporation of the STandard for the Exchange of Product model data (STEP) standard into our resource ontology has already been shown in section 4.6.1. Here, we introduce and show how to match another standard named ASAM ODS.
ASAM is an incorporated standards association, especially of the automotive industry. The vision of ASAM is to optimize and establish seamless data interchange and interconnection between products (ASAM 2006). The consortium develops, maintains and deploys platform independent extensible standards for the "measurement, automation, analysis and simulation systems used within industry, and to support software engineering methods" (ibid.). These standards have been adopted by many large manufacturers, like Volkswagen AG, General Motors, Daimler AG, BMW AG and Audi AG. One of these standards is the ASAM ODS.
ODS is "that part of the ASAM standards which focuses on persistent storage and retrieval of data" (Bartz 2009). One aspect of this standard is the depicted data model in figure 7.12.
The ODS model is separated into three layers (ibid.): the base elements seen in the figure acts as a meta model for an application model. The application model comprises all the entities of a real application. The standard provides application models for 'Noise, Vibration, Harshness' (NVH), calibration data and others. One application model, the workflow application model, is used to describe complete

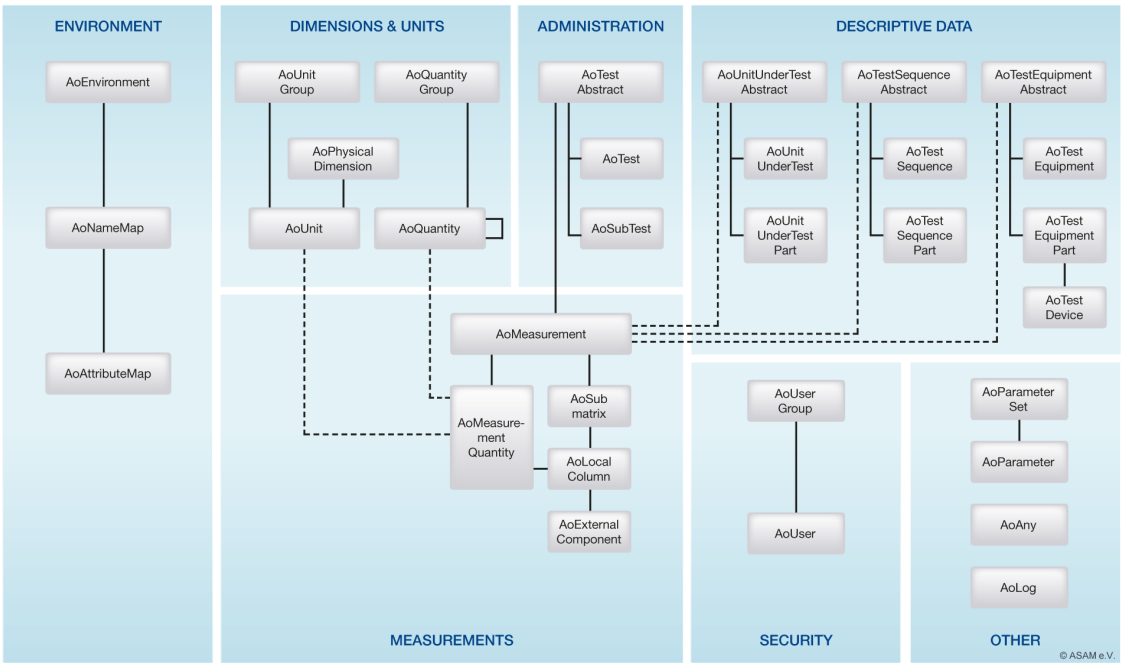


Figure 7.12.: The ASAM ODS base elements (Thomsen 2013).

specifications of processes in order to achieve a result. It allows the description of sequential and parallel processes by adopting concepts known from Petri nets. The basic principle of such an application could also be realized using our method ontology as a model and a data flow analysis tool or a BRMS. The case study PDD in section 7.5 demonstrates, how various methods can be linked in a sequential chain in order to produce such a desired result. Besides, our method elements are linked to processes by representing an action in a business process and various tools and standards exist in order to make processes executable.

The final ODS layer is the instance layer. It stores information about measured values and describes the application elements with data values, e.g., the descriptive data of a tested engine.

In order to align our method model to this standard, we focus on the base elements, because we also provide a meta model to describe the method information. Elements described in an application model can be aligned to the entities modeled in the method ontology's ABox.

Because our method ontology is not specifically meant to describe CAT test configurations, but methods in general, we can only reuse a subset of the base elements. The alignment between this subset is exemplified in the following which is also depicted in Figure 7.13. The group *Descriptive Data* best reflects our method definition. The *AoTestEquipment* is a subclass of our `Tool`. Tools can either

be represented by software applications, devices, implements or equipment, like a test bed. They describe with which equipment a method can be performed. *AoUnitUnderTest* contains information on what has been tested. This is similar to our class *Resource* that represents an object a method can be applied to. The *AoTestSequence* describes the steps that have been processed when testing a unit. The steps that are necessary to perform a method are described in the method definition as well. In our method ontology, a sequence can be expressed by relating different `Procedure` instances with the according semantics (*has_predecessor*, *has_successor*,...). All the elements of the *Descriptive Data* group can be further partitioned into sub-groups, i.e., *AoUnitUnderTestPart*, *AoTestSequencePart* and *AoTestEquipmentPart*, enabling a more detailed and finer-grained representation of the elements.

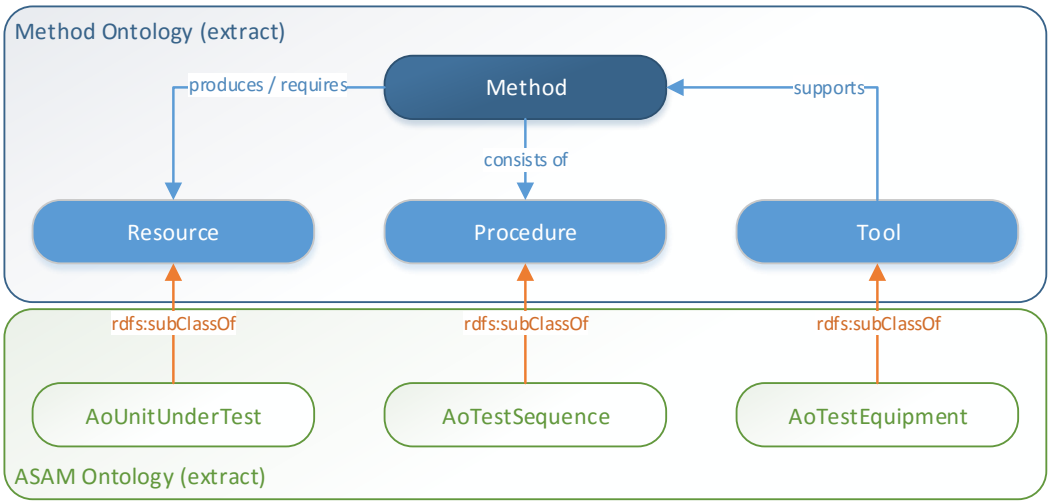


Figure 7.13.: Possible mapping between ASAM ODS and Method ontology (extract).

The other groups depicted in Figure 7.12 offer elements to describe test data in more detail. The *Measurements* group is used for storing the location of the processed data. This information is not directly useful for our method ontology, because we are not representing low-level data elements but a more abstract view on the entire enterprise's methods. The remainder of the groups are used to organize tests in projects, describe units, for instance, SI units, and quantities or offer security management for the users. The group *Other* contains elements that can be used to describe any other data (*AoAny*) or logging information. They have no direct connection to the definition of our method ontology. An exception is

the group *Environment*. The elements *AoNameMap* and *AoAttributeMap* provide the ability to define aliases for entries in the application model. This feature can easily be realized in the method ontology using either OWL labels with the according language annotation or by making use of linguistic ontologies like SKOS (see section 4.6.3). Using SKOS even provides far more possibilities when describing ontological elements, like stating a preferred and alternative names for an object. More details about the various remaining ODS groups and elements can be found in the standard specification (Bartz 2009). Nevertheless, the whole ASAM ODS base model can be mapped to an OWL ontology, like demonstrated in (Marcos et al. 2005), and thereby, our method ontology can be extended and aligned as necessary or more precise, used as an upper ontology that can be customized by a domain ontology as depicted in section 4.6.

Besides the common data model, ODS consists of interface descriptions, a database model and application models, like workflow descriptions, NVH data or crash test data (ASAM 2013). Another aspect of the ASAM ODS standard is the data exchange format ASAM transport format (ATF). ATF is a file-based syntax for the exchange of data between software applications.

Based on this data model, Audi created an own data model in order to store measurement data of various software programs (*cf.* ASAM 2006, p. 18). It describes the base model elements used in CAT for testing and measuring test specimen, i.e., vehicle components and sub-projects. This data model includes a model element `measurement`. Together with its context information, it describes how a measurement is to be conducted: "This is a list of measuring points, data channels, the sensors used and the parameterisation of the measuring methods" (ibid.). A `Method` combines the procedures that are conducted to produce the results of this measurement. Attributes, like weight or weather conditions, are included in the method definition. The method describes "with what" a test can be conducted, i.e., the tools used (test bench, system), the measurement itself (the "how") and the specimen it is applied to.

When comparing the standard ASAM ODS model and the specialized version, introduced here, many commonalities with our method model can be observed. Their methods produces some kind of result, which is in turn an input for a succeeding analysis. These relations are considered in our model as well. More important, their intrinsic method definition consists of the same elements: the measurement describes how a method is to be conducted – the procedure – and the specimen represents the unit that is to be measured. In our model, an analy-

sis is not part of a method, but a process that can be conducted with the help of a method.
However, when they created their model, they had a different point of view regarding methods. They created the model in order to harmonize and unify the measurement data of the various software programs that are in use at the company. Our approach is concerned with the description of semantic knowledge, annotations and analyses regarding these methods, not the method measurement data itself. Nonetheless, this data model validates our work and reusing parts of their model for our method description allows an easy integration of measurement data in the future.

7.5. Comparing Methods in Property-Driven Development

7.5.1. Introduction

One of the main goals for the development of this demonstrator is the attempt to conserve business and domain knowledge separated from program source code. This way the business knowledge becomes reusable in other projects and is maintainable by the dedicated user roles.
Regarding the business case, the demonstrator is meant to support engineers and other stakeholders by elucidating and visualizing coherences of CAx methods, product information, process steps and vehicle properties throughout the car PDP. Therefore, our KB resembles the method ontologies introduced in chapter 4. By applying our methodology described in the previous main chapters and hence storing the knowledge in ontologies combined with rules, we plan to achieve the attempted goals. Furthermore, we integrate Enterprise Architecture Management (EAM) concepts into our KB.
The focus and contribution of this section is the formalization and modeling of this knowledge in ontologies, rules and queries, the architecture, its analysis, evaluation and the KM approach behind these endeavors.

Nota bene

An initial description of this case study has already been published in the ONTORULE project (Rosina and Kiss 2011; Kiss et al. 2010), along with alternative solution approaches. Therefore, this case study is based on this publication and reuses parts of the deliverable, either directly or indirectly. The originated demonstrator from this case study has been implemented as a web application and has been published online[5] together with a significant example to showcase a proof-of-concept. The web application acts as the interface to the end-user, for instance, DEs. It queries a rule- and ontology engine, to visualize the underlying explicit and deduced knowledge.
The business case is based on a methodology called PDD (Weber 2005), an approach to validate properties, e.g., safety or driving comfort that the product has to fulfill, and the analysis of information chains for the product development (Westphal and Wartzack 2010; Westphal and Wartzack 2011). This business scenario has been jointly gathered by domain and IT experts.

Outline

In the next subsection 7.5.2, we explain the use case in more detail and introduce some vocabulary used. The following part of this introductory subsection deals with the requirements, comparing the current and targeted situation. Furthermore, we depict the expected benefits of our approach and present the involved user roles.
The solution approach for the demonstrator is presented in subsection 7.5.3. It includes a part about KA, modeling and design.
In subsection 7.5.4, we describe the evolved solution, i.e., the implementation, to these requirements, illustrating the demonstrator's architecture and the underlying data model. Additionally, the demonstrator's functionalities together with its UI components are illustrated in detail. Afterwards, we introduce the underlying ontologies, next to the rules and queries. The appropriate example screenshots and listings have been attached in the annex of this thesis. You can find them in chapter B.
Subsequently, the demonstrator is evaluated in subsection 7.5.5 which includes, among others, a comparison to the requirements and DE interviews.
Finally, we conclude the case study in subsection 7.5.6.

[5] http://ontorule-project.eu/demonstrator-m32/

7.5.2. Property-Driven Development

One of the initial steps in the PDP of a new car is to define target properties, i.e., specify features that are required and experienced by customers, like driving comfort, safety or sportiness. These target properties are listed up in catalogs, consisting of different detail levels describing the cars desired, required and mandatory behavior in various granularities, e.g., "braking" is a sub- or *level-2*-property of the *level-1*-property "driving comfort".
These requirements are detailed during the concept phase in order to create the requirements for each component and part within the car. The concept phase ends, when the developers are confident of fulfilling all requirements with the planned components and parts.
In the next step, the engineers start to design new or modify already existing components and parts in a so called micro cycle. They design a `Solution Concept` consisting of geometry as CAD models, software code or connection schemes.
Later, this `Solution Concept` is filtered, preprocessed and assembled to analysis models during the model creation step. In the following step, the product properties are analyzed in order to get the actual properties virtually (CAE) or physically (CAT).
In Figure 7.14, we depict the use case in a high level ontology schema: processes (`Process Information`) use different `Product Information` and methods (`Method Information`) to safeguard the vehicle's properties (`Property`). These methods can either be `Virtual Methods` or `Physical Methods` and are using `Tools` in order to be performed.
Besides, the case study's ontology shall be able to distinguish between product properties and characteristics (Weber 2005).

Definition 7.1 (*Property*)

"Properties (P_j) describe the product's behaviour, e.g. function, weight, safety and reliability, aesthetic properties, but also things like manufacturability, assemblability, testability, environmental friendliness, and cost. They cannot be directly influenced by the developer/designer" (Weber 2011).

Definition 7.2 (*Characteristic*)

A product characteristic describes the product's measurable features, like its material, dimensions, shape, structure or surfaces (ibid.). They can be accessed, created or changed by an engineer or designer (Westphal, Meerkamm, et al. 2009; Weber 2011).

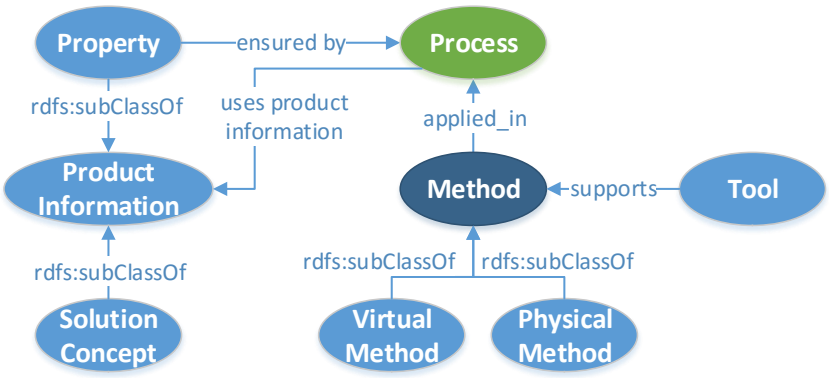


Figure 7.14.: Simplified extract of the case study ontology.

For example, the calculation of the *system pressure* of an air supply aggregate requires a model of the *piston force* and the *piston area*. This *piston area* can, for instance, be preprocessed out of the *geometry* of a CAD-model. And finally, these actual properties are compared with the target properties. This assessment decides if the next step of the macro process can be executed or if a further optimization loop or even a modification of the requirements has to be performed.
Methods and tools are used to perform each of these process steps. The entirety of these CA attempts and approaches, physical and virtual, is called CAx methods. For example, new electronic components, like an *Electronic Stability Control (ESC)*, are tested in *Hardware in the Loop (HiL)* simulations that make use of virtual models that behave like the related dynamic systems.
But the validation of product properties has to be done along the product hierarchy; whereas the solution concept of a top layer defines the requirements for the next, deeper layer.
For example, the property *driving comfort* of a vehicle (product) requires a special *suspension* (assembly) as solution concept (Herfeld 2007) (*cf.* Figure 7.15). This *suspension* requires a distinct *damping ratio*, which should be realized with an *air damper* (sub-assembly). This *air damper* requires again a distinct *system pressure* of the *air supply aggregate* (component). Therefore, the micro cycle has to be pro-

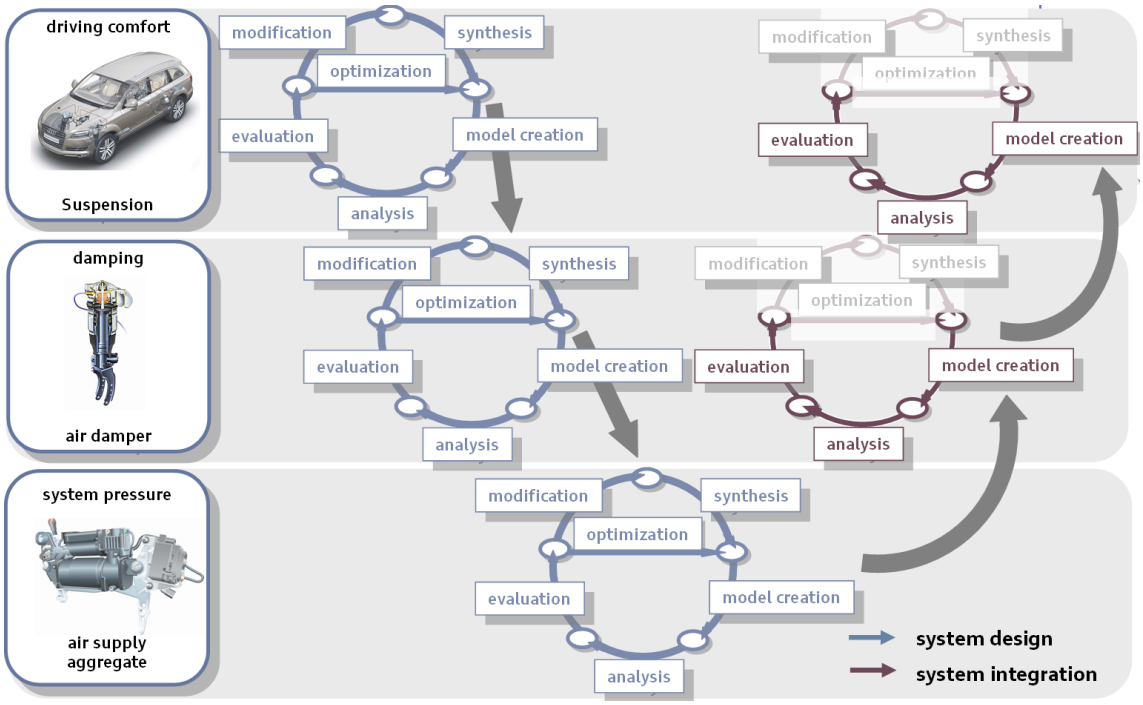


Figure 7.15.: Process model of the product property validation (Rosina and Kiss 2011).

cessed on every layer of the product hierarchy until the solution concepts fulfills the requirements. These requirements are deduced from the solution concepts that are one layer higher in the product hierarchy. The bold, gray arrows represent the transition between the hierarchy layers.

During detailed design, each CAx discipline has to create solution concepts for their components. Finally, newly created solution concepts are tested and assembled to sub-assemblies, assemblies and to the product in order to evaluate the properties and functions during system integration. But unlike the system design phase the system integration phase has no synthesis, modification or optimization steps, because if a property is not fulfilled a drawback to the system design phase is necessary. Consequently, it can be postulated that the macro process of the PDP is a distinct sequence of the micro cycles. These micro cycles are executed in different layers of the product hierarchy. Last but not least, it is necessary to assess an ability to each process step. The ability should be described in this context by the four types of Key Performance Indicators (KPIs) – costs, time, maturity and quality, which resemble our introduced quality attributes from section 4.4.3. The actual ability depends on the used methods or tools because these are used in every process step to transform input information into output infor-

mation. For example, an analysis step (crash analysis) transforms the ingoing test scenario and the analysis model into an outgoing assessment criterion, e.g., the Head Injury Criterion (HIC) – a criterion for the severity of an injury after a vehicle crash. By using a specific method in combination with a software tool for this analysis, the process step has a distinct expenditure of time and costs and the outgoing assessment criteria have a distinct quality.
Consequently, the consideration of the used methods and tools allows an allocation of quality to each product information and an allocation of an expenditure of time and costs of each process step.

Requirements

The knowledge about the initial relation between a target property that was defined in the beginning of the PDP and the solution concept with its related CAx methods often perishes during the progression of the vehicle's development. In addition, the requirements and target properties may change over time and demand new solution concepts, new virtual models and prototypes. Consequently, an effective planning and monitoring of the product property validation process as well as the effective support of a change management requires an integrated model of the required properties, the used methods and tools, the created solution concepts and their procedural connection.
By modeling the described scenario in an ontology, we expect to reduce the knowledge gap between the various process steps for the involved employees and departments. Sharing the knowledge by using a common tool, standards and data basis will reduce time-consuming data acquisition and ensure that the involved personnel access identical data, which will help to speed-up the development and innovation cycles. Moreover, the ontology will help to analyze and optimize the actual processes and used methods. Besides, the modeled knowledge can be reused in succeeding applications.
One of the goals of this case study is to support the DEs by determining the information chains from the rating of properties to the underlying solution concept via the involved methods and other concerned analysis steps.
Furthermore, the combination of sundry virtual and physical CAx methods requires different costs and time spans to produce miscellaneous qualities. For example the execution of a CAE simulation may take only minutes. However,

the preparation and modeling may takes weeks. These indications will support managers when planning their projects.
Besides, the system can be used to compare different method and information chains at different development stages. This will help the responsible manager to find processes which are cheaper, faster and even offer a higher quality. The anticipated reduction of the development costs is caused mainly by finding virtual replacements for physical analysis chains, which are less time-consuming and require less material. Moreover, in the ontology the different analyses are linked to their required IT and physical tools. They will help IT managers to plan and illustrate the enterprise's IT architecture, e.g., by visualizing the workflow.
Besides interviewing DEs, a lot of knowledge in the intended ontology originates from various documents like regulations, laws or internal documents. By linking ontology instances and relations to their source documents, which are also part of the ontology, the users can easily confirm and verify the properness of the modeled semantic relations and the used linguistic terms of the various entities.
The knowledge acquisition for the case study was the most comprehensive step in the development. Together with DEs we defined the requirements and collected information from various sources.
With the development of the described business scenario we intend to be able to validate and safeguard vehicles' properties earlier in the PDP by finding the most appropriate methods considering quality attributes like maturity, cost and time. To achieve this, the demonstrator should be able to support the planning and monitoring of the process of product assessment, analysis and model creation. The flexible design allows further enhancing the demonstrator with more competitive priorities (Hayes and Wheelwright 1984; Foo and Friedman 1992) in the future. For example an additional ontology describing the concept design can be added.
The main business questions that the system shall be able to answer are listed up in the following:

- What is the earliest point of time to assess a product property in the PDP?
- Which methods are applied?
- Which resources are necessary as an input for the method(s)?
- What is the forecast KPI of this product property validation?

In order to achieve these goals, we will model the information flow within the product property validation process. That means, we have to formalize the busi-

ness logic in a suitable format and integrate the already available knowledge, for example legacy data from existing spreadsheets. This also applies to the linking of documents and data managed in file systems or data management system to the instances of the ontology.
The structure and the contents of our KB have to be maintainable and we want to be able to validate its correctness, because new information is emerging permanently and existing knowledge is prone to changes. Thereby, we have to consider that sundry DEs of different departments manage the underlying information. They know the BRs, how the different process steps, product information and CAx methods are connected to each other and when which one is used.
Together with the IT architects or business analysts, they design the model and formalize the rules.
Our KB can then be analyzed along its information chain — including product, process, functional and IT information — from the product property to the CAD-model of a component. Besides, we want to analyze the covered methods and tools as well the generated quality, cost and time. This means, that the technology used must allow defining criteria for method comparison. Additionally, we also require a comparison of alternative validation or safeguard chains relative to the used methods and tools.
Furthermore, we want to analyze the coherences between the product information and the according data management system. We also want to know which product is used in which process step which again shall link the according organizational unit.
Next to the business requirements, we also have some requirements regarding the demonstrator's usability and functionality resp. its UI. Various user roles are involved as end users in the scenario described in this section. Each one needs to access different parts of the KB in order to fulfill its role specific tasks. For example, a project leader requires a validation of the target property profile and a check of its completeness. Different DEs require the KB in order to understand and visualize interconnections of their work and processes. Method and process managers, on the other hand, have to optimize the deployed tools, enhance existing methods or develop new ones where they see fit.
Therefore, the demonstrator shall support the sorting and plotting of the earliest possible product property validation method against the PDP timeline. Furthermore, each validation chain should be separately visualized for the best qualitative, cost and time effective solution. It shall present a detailed visualization of

the used methods and the total quality, costs and time for each milestone within a property spider. Besides, we want to depict a detailed visualization of each validation chain including the product-, process- and method-information. Finally, we need a visualization of the necessary component data and information for each property validation.

7.5.3. Approach

During the KA, as well as the operation and evaluation phase, the end users, typically the DEs, usually just correspond with the KEs, for instance, enterprise architects. Other stakeholders of such an application, like managers, BRs or governance experts, should also correspond with the KEs, because this role fits in best between domain and IT knowledge. The KEs communicate with the programmers of the IT application and discuss the interfaces to the required queries for providing the needed UI elements and interactions. This way, DEs can create or manipulate facts in the KB on their own, without the need to discuss every change with other roles involved. When making changes to BRs, queries or the structure, i.e., the TBox, they work together with the KEs, again.
Following the described procedure, the business requirements for this case study originated directly from DEs, i.e., engineers, in the product development at Audi R&D, by performing interviews and thus elaborating a conceptual model. Together, we initially discussed, specified, expanded and discarded sundry aspects of the challenges which ultimately resulted in a jointly agreed requirements specification for this case study.
Already during the requirements elicitation, we used ontologies as a conceptual model for discussing various aspects about class relationships and properties. BRs, documented in natural language, further helped discussing the complex characteristics of this case study with the DEs.
Besides, we also analyzed documents about the case study, e.g., regulatory documents about the method execution, and performed data table analyses. In the KA phase, we also made use of different NLP methods in order to extract ontologies and candidate rules from regulations (Omrane, Nazarenko, Rosina, et al. 2011). In addition, we applied methodologies of expert-aided modeling to come up with parts of our ontology and validate it's correctness (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012).

These information have been the basis for creating the TBox that has been modeled in OWL. As a result, ontologies have been designed and populated, that reflect the coherence between the various product information, process steps, properties, CAx methods and the supporting IT. The population data primarily originated from the import of existing structured data, i.e., databases and spread sheets.
Furthermore, the queries required for the use cases and demonstrator's views, together with their underlying BRs, have been defined in detail. The OWL ontologies, rules and queries had been transformed into ObjectLogic, because we decided to use this language and the corresponding tools to author, maintain and execute our KB. Following the suggestions of the SW Stack, implementing the BR with RIF would have been plausible. However, at the point of time of the prototype development, RIF had not been mature enough. For instance, the language did not yet support expressing aggregates. Furthermore, the tool support was insufficient.
So we incrementally expanded, adapted, populated and tested the resulting ontologies, together with the associated rules and appropriate queries using ObjectLogic.
Simultaneously, we charged an IT expert with the development of the prototype UI, based on the devised query results and collected requirements. After performing several tests and improving the developed prototype, the underlying KB and their connection with a limited number of end users, we evaluated the application in sundry R&D departments, presenting our solution to different roles and performed interviews. Besides, we performed usability tests together with usability experts.

Nota bene

In the course of realizing this case study, we developed two different ontology and back-end solutions. In a first iteration, the feasibility of this project has been verified with a small data set and only a limited number of rules and possible queries.
This initial proof-on-concept version has acted as the origin for the resulting two different versions: the complete ontologies that has been tested consisted of nearly 130K facts, resulting in huge and thus long query answers, because

the underlying rules created a lot of permutations for creating the macro and micro cycles of the *method chains*; ultimately leading to combinatorial problems. The ontologies created for the demonstrator consisted of noticeable less entities compared to the entire ones. Only the *level-1*-property *02_Driving_behaviour* and its *level-2*-property *02_04_Braking* have been included in this version. Nonetheless, this limited selection is sufficient to demonstrate the whole functionality. In common with the entire ontology, the ontology consisted of 39 classes with their aggregate number of 60 properties.
We had various reasons for developing different solutions.
Firstly, the complete ontology consisted of authentic data that is confidential and hence cannot be published. Another important step in the development of the extended ontology, resp. the architecture, was the connection to existing databases. We mapped different concepts to database tables that existed in the IT landscape and their columns to the related ontology attributes. By promoting this approach we did not need to maintain facts at two different places and could concentrate on maintaining the TBox, rules and queries. The management and maintenance of the TBox, rules and queries is the task of the business analyst or KE. The entities, i.e., the ABox, is fed by various existing systems that are maintained by the responsible DEs.
Secondly, a small, meaningful, but distorted, extraction is easier to handle and the reasoner behind this use case's architecture can work much faster when querying less data. Furthermore, we created a copy of the relevant ABox extract from the databases for the demonstrator's ontology. This guarantees repeatable, testable results and allows the demonstrator to run independently from the internal IT infrastructure.

Knowledge Acquisition

Populating the ABox was one of the major challenges during the development of the demonstrator.
During the creation of the business ontology for the prototype we developed a methodology and belonging tools to convert information stored in Excel files into OWL ontologies. This was our primary procedure of populating the ontologies which resulted in several thousand facts. Other types of data acquisition were mapping the ontology concepts and attributes to database tables and columns using built-in functionalities of OntoBroker. Besides, OntoStudio provides mech-

anisms to load spreadsheet files directly into an ontology project and link the relevant data rows and columns to ontological concepts and properties. However, the anticipated structure has been incompatible with our sources.
The remainder of the facts have been added manually by DEs and KEs alike.
For importing the data from Excel sheets, the original way of storing data within this case study, we implemented an auxiliary rudimentary tool that supported us in the acquisition process. The tool imports files of a specific format (*cf.* Figure 7.16), namely a matrix representing an $N : M$ relationship between two database tables. Covering all the possible structures in Excel spreadsheets is not possible because of the varied layout and structuring possibilities. Therefore, we decided to convert our heterogeneous data sources into an identical structure. The type of customization that is successfully processable with our extraction tool is straitened to a varying number of attributes and instances, but only converts spread sheets that represent a class-class table. Therefore, a source table consists of an arbitrary number of attributes, instances and relations to another table's instances. The tool expects the user to provide an OWL file to populate, the Excel file and the selection of the appropriate OWL classes. That means the OWL file including the classes must exist beforehand. The attributes, resp. the data properties, are added automatically with their correct data type if it is obvious. Otherwise, in the case of ambiguity, the user is asked to specify the type of the to be created data property.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		Attribut1_1	Attribut1_2	Attribut1_3	Attribut1_4		Instanz2_1	Instanz2_2	Instanz2_2	Instanz2_3	Instanz2_4	Instanz2_5	Instanz2_6	Instanz2_7	Instanz2_8
2	Attribut2_1						4	5	5	6	34	3	5	8	6
3	Attribut2_2						true	false	false	false	true	false	false	false	false
4	Attribut2_3						TGZ	ABD	ABD	TGZ	ABD	HHH	KLJ	TGZ	ABD
5															
6	Instanz1_1	5	T	3,40	5		bewertet_durch				bewertet_durch			bewertet_durch	
7	Instanz1_2	4	R	2,50	4				verantwortet	bewertet_durch					
8	Instanz1_3	3	T	4,30			bewertet_durch	bewertet_durch	verantwortet		bewertet_durch		bewertet_durch		
9	Instanz1_4	6	Z	34,20	3		bewertet_durch								
10	Instanz1_5	5	R	4,60	5		bewertet_durch		verantwortet	bewertet_durch		verantwortet	verantwortet		
11	Instanz1_6	4	J	2,00	3										
12	Instanz1_7	3	R	1,00				bewertet_durch							
13	Instanz1_8	6	E	2,00	2										bewertet_durch

Figure 7.16.: Example of an importable Excel sheet.

The tool asks if the new information shall be added to the existing file and if the data can be overwritten.
Most of the existing spreadsheets that we analyzed during the knowledge acquisition phase could be manually converted to fit the above described structure. This was a time-consuming, but worthwhile approach, resulting in ontologies covering more than 100K facts.
With these procedures, the OWL ontologies, that are the basis in the internal and demonstration application, were populated.

Modeling and Design

The acquired knowledge now had to be modeled and formalized in order to execute it within our prototype application. As described, this was an iterative approach where we collaborated tightly with DEs. The conceptual models, together with the OWL ontologies worked as basis for the TBox, that has been realized with the ontology development tool OntoStudio 3.1 in order to support ObjectLogic. Thereby, OWL ontologies can be imported and translated automatically into ObjectLogic projects. As a result, we came up with the final model depicted in Figure 7.17. The schema displays the complete TBox from all the imported modules, which are described in more detail in the following pages and in section 7.5.4. Furthermore, it illustrates the rule-based relations that are created by reasoning.

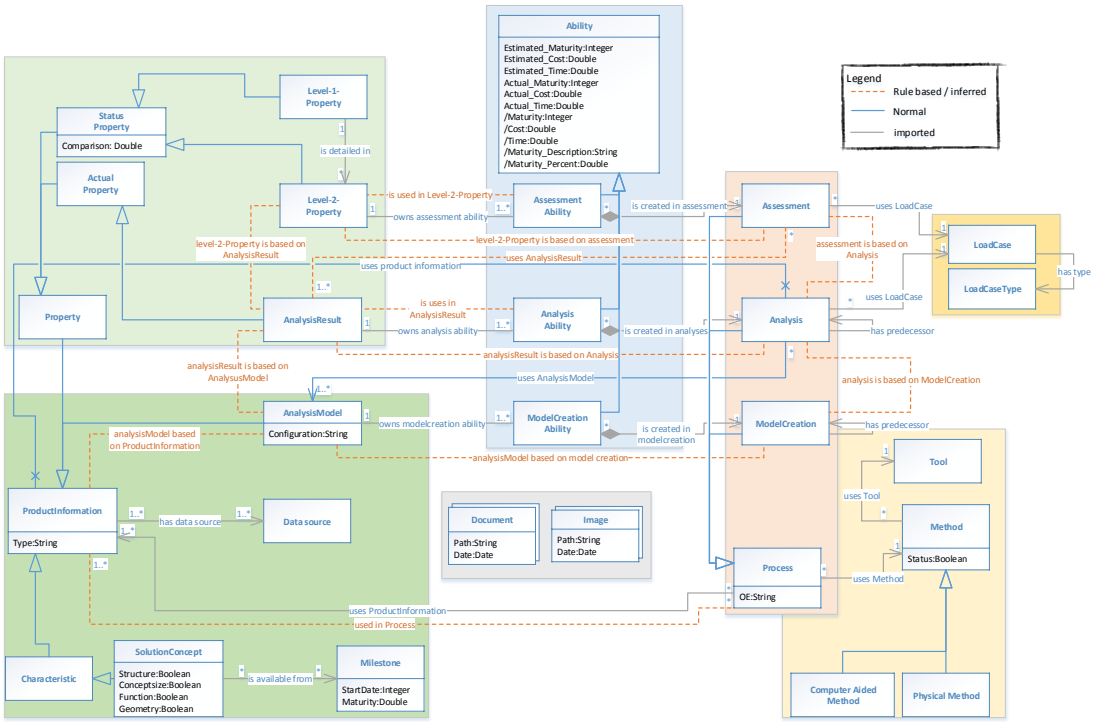


Figure 7.17.: PDD Schema.

The hallmark in our ontology is one of the `Processes`, i.e., `ModelCreation`, `Analysis` and `Assessment`, that can be found in the salmon-colored area. These processes connect the used `ProductInformation`, represented by the green area, with the applied `Methods` (sand-colored box) and the resulting `Property` in the light green area via the defined KPIs, i.e., `Ability`, in the light blue-colored center of

the diagram.
The ProductInformation consist of Properties and Characteristics, which represent SolutionConcepts and their instances that describe real or virtual parts, sub-assemblies and assemblies of a car. A SolutionConcept can either represent a function, geometry, concept size or structure information and is related to the class Milestone. The data property is_available_from can be used to derive the dependent methods' possible execution time span, i.e., the earliest StartDate in combination with the time data property of its Ability in order to return the results on schedule.
Also, the Tools used to perform a method are modeled in our ontology. These tools can either be software tools for CA methods or physical tools, like a braking rig for prototype-based methods. Additionally, there are two minor areas: the background information, like Documents and Images (gray area) that can be linked to each entity and the LoadCases in the yellow box that are connected to various process steps.
A more fine-grained description of all the involved classes can be found in section B.3.
During the modeling we also considered the multilingualism of our application. By adding *_label* constructs in German and English to the entities in the ontology, they could be represented in either language in the prototype application.
The correctness of the envisioned business model solution has been continuously validated by the implemented consistency checks (Korf, Kiss, Durand, et al. 2010; Fink 2011) that have previously been formulated by the interviewed DEs. These rules have been implemented in our test module (*cf.* section 7.5.4) in order to check the model's integrity. They allowed us to find and correct orphaned, redundant, falsely related and missing ontological entities appropriately.

7.5.4. Implementation

Application Architecture

We decided to implement the prototype as a web application, because this way, it can be accessed from any browser without additional and usually costly installation procedure and as a consequence, our KB as well as the application are

up-to-date all the time. The demonstrator has been implemented using GWT[6] with some extensions and is deployed on a Tomcat server[7], running on a web server. In Figure 7.18, sundry involved devices and their connections are illustrated.

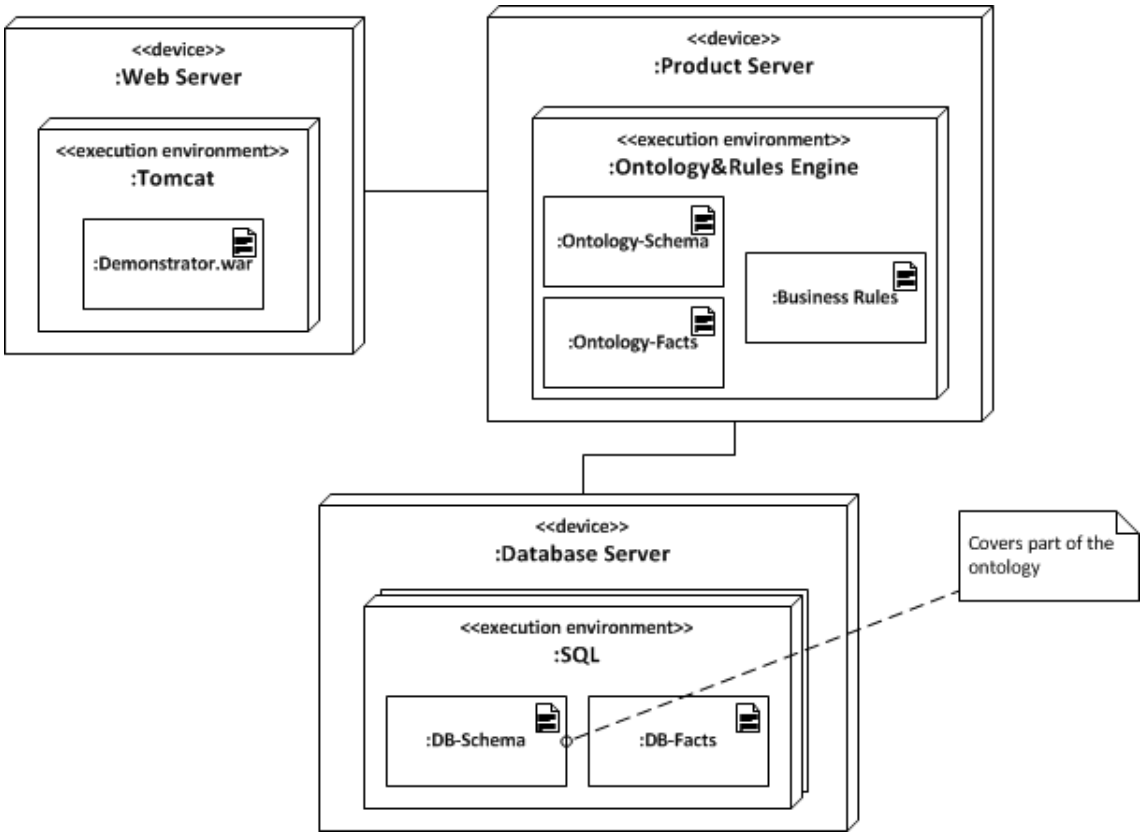


Figure 7.18.: Architecture.

Rule, ontology and inference engine in this architecture is OntoBroker 6.1, which supports the knowledge representation ObjectLogic. Ontology schema, part of the instances, rules and queries were deployed on this SW middleware. The GWT application called the predefined queries by name, thus, we deployed as little as possible business logic in the application layer. As described in the approach, we integrated databases by using OntoBroker's built-in functionalities. The database tables have been mapped to our TBoxes, thus populating the ontologies. Besides, the majority of the ABox has been populated by storing the facts directly in the ontologies.

[6]See http://www.gwtproject.org/ for details; last accessed 10/11/2014.
[7]Apache Tomcat: http://tomcat.apache.org/

Ontology Architecture

The ontology which we developed to reflect the data model of our system has been structured into *modules*[8], which are imported to a main business ontology ("*Master*") containing rules and queries for the views.

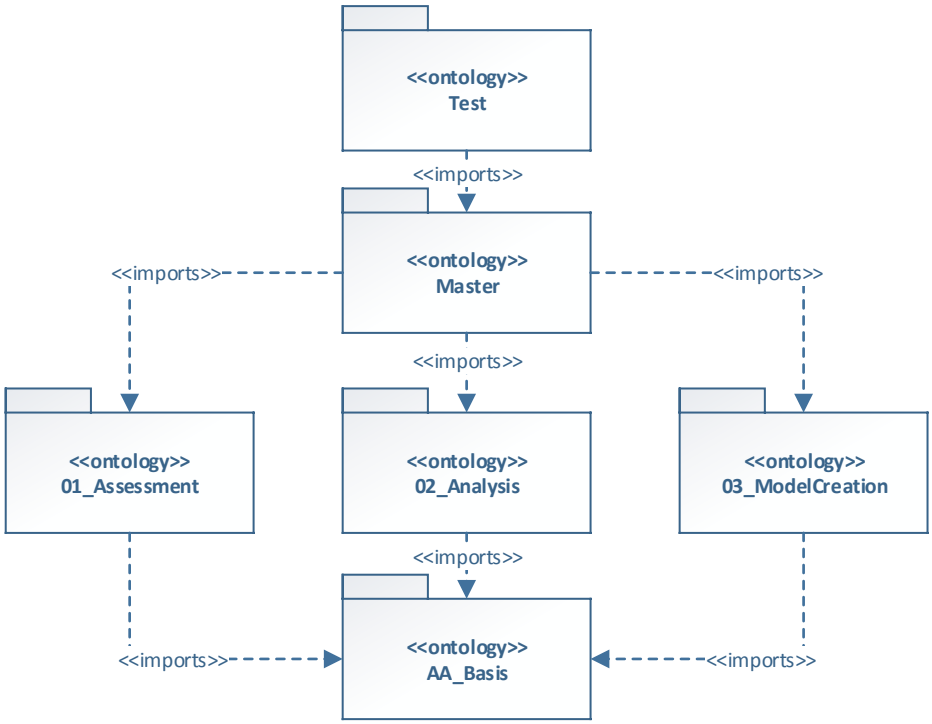


Figure 7.19.: Ontology Import Schema.

An overview of the schema and import relations is shown in Figure 7.19. A more in-depth explanation of the modules can be found in the following subsections and the appendix section B.3.
The top-level concepts and their relations that are used by every other sub-module are part of the *AA_Basis* ontology. This module is imported by *01_Assessment*, *02_-Analysis* and *03_ModelCreation*. Each of these modules represents another process and the connected product information, qualities and other information. They are combined in the *Master* module, which contains the overall schema together with the rules and queries.

[8]In ObjectLogic, the different ontologies are called "modules"

The *Test* module on top of the *Master* module is, like the name suggests, just for testing purposes during the modeling process, rule refinement and maintenance. Since the entire master module is imported, Create, Read, Update, Delete (CRUD) operations can be applied to all queries, rules, TBox and ABox entities in this module and hence tested and evaluated before committing the changes to the master module.

Demonstrator method architecture artifacts and graphical UI

The prototype application that we implemented comprises many functionalities and hence artifacts and UI views for the different end user roles. In this subsection we will introduce the various views, explaining the intended operability. Please note, that only the landing page is displayed in this section for an improved reading flow. The remaining UIs are located in the annex in section B.1.
First, we will clarify the general elements that can be rediscovered on every page on the basis of Figure 7.20, which represents the web application's landing page with an already loaded car project. The navigation bar at the top shows different tabs and buttons that lead to the respective analysis functions of the business scenario.

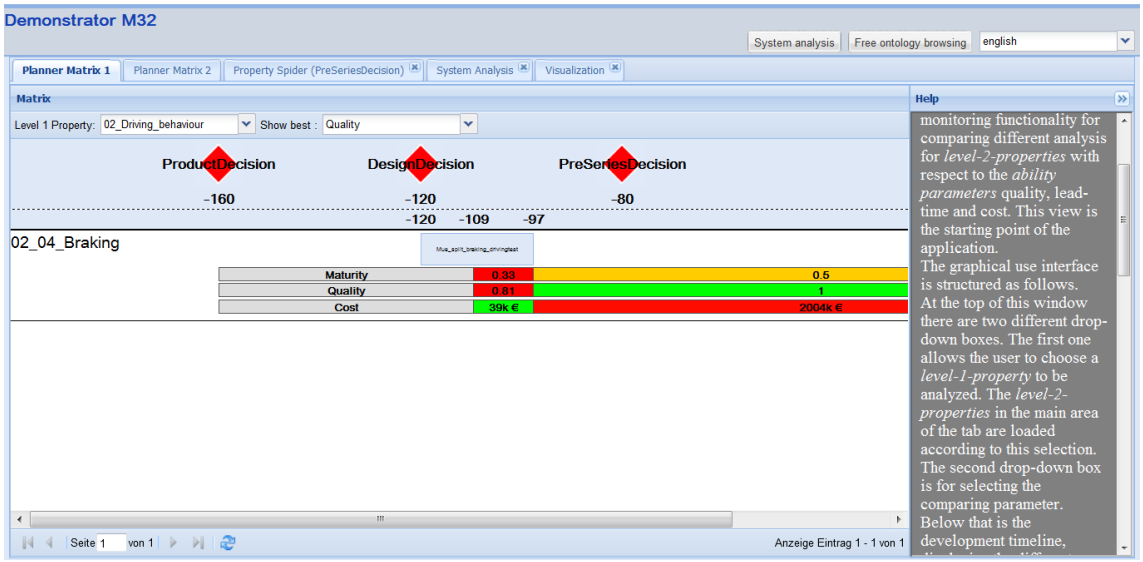


Figure 7.20.: PDD demonstrator landing page.

In the upper right corner, the application's language can be changed, i.e., labels, menus and help language, which also changes the language of the visualized on-

tological entities. The default language is German which is the fallback, if no translated term in the selected language could be found.
Below the top navigation bar are the currently opened tabs, which can be switched, closed and changed on demand. On the right panel of the main content a context-aware help section is displayed.
Inside one of the *planner matrix* tabs (Figures 7.20 and B.1), the end user can compare different analyses for *level-2*-properties with respect to the KPIs maturity, quality, time and cost. The presented method chains can be sorted and filtered according to the user's needs. At the top, the chronologically ordered milestones indicate the PDP – all methods, resp. products, from the begin of the PDP to Start of Production (SOP) are related to a milestone, along with a time unit (*here:* weeks till SOP). Clicking on one of the milestones leads to the milestone's *property spider*.
The core functionality is the diagram spanned by the milestones and the *level-2*-properties. The lengths of the bars represent the time consumption of the analysis steps, i.e., method chains. At the bottom of each row are three different, colored smaller bars, indicating colorized values for the KPIs — from red=bad/expensive to green=good/cheap, along with all the intermediate colorization steps. If the user needs more in-depth information of a specific method chain, a click on its label will lead to the quality path view.
The *property spider* (*cf.* Figure B.2) indicates the target property values of the loaded car project. We decided to display two different layers, *level-1*- and *level-2*-properties in this demonstrator. Actually, the properties can be fanned out to even more detailed and deeper layers.
Selecting one top-level property leads to a lower-level spider. In the left part, pictures symbolize the various properties to simplify the navigation and offer links to further analyses.
The *information chain* page (*cf.* Figure B.3) displays all product information and related process steps for the selected *level-2*-property and milestone. Additionally, the methods, tools and loading cases are displayed for a detailed overview.
The *product information* page (*cf.* Figure B.4) illustrates a table of all the required product information including their related attributes for the selected *level-2*-property and milestone.
In the *quality path* page (*cf.* Figure B.5), the connected instances that are included in the process step are visualized. This view focuses on the related KPIs, presenting a detailed view of each method involved in the method chain along with the relevant ontology extract. Each method is displayed with the connected tool and

its single costs, time consumption, maturity and quality.
The *system analysis* view allows the user to query specific information from the ontology that is displayed in a filterable, sortable table, depicted in Figure B.6. The user can choose to view the relation between tools and product information and which product information is not managed by any tool. This could either be a missing relation in the ontology itself or a hint that the information itself is not available at all. Furthermore, all relevant knowledge for a selected process can be shown, i.e., its name, responsible organizational unit, connected product information, test scenarios and method information. Finally, a list of tools together with the related methods and a status whether this method is already in use or is planned for the future.
The final supported demonstrator view is the *free ontology browsing* page seen in Figure B.7 which allows the user to browse the whole ontology in a convenient way. If some information cannot be found or deduced by the existing pages and functionalities, the user has the option to search for the required connections in the ontology itself.
The instances of concepts and their relations are plotted in a view to visualize their connections.
By clicking on one of the instances in the right panel, the user can change the focus to this very item and display *n* relations in every direction, depending on the selection of the slider *"Dependencies"* on the bottom left of the right panel. The instances and attributes are spread and arranged automatically to ensure a good overview. Depending on the selection of the slider at the bottom center of the right panel (*"Node distance"*), the distances can be adjusted. All the instances are colored according to the selection in the tree-like view on the left.
Above the sliders, a history shows the path of the selected items.
This view offers many options to filter, rearrange or highlight the required information. On the left is a tree-like view of the ontology's classes and properties, which can be expanded or collapsed on demand. The hierarchy is derived from the super- and sub-concept relations. By (de-)selecting the check boxes, the information shown in the right panel can be filtered. The icons on the lower left are used to pause and resume the automatic weighted spreading function for the ontology views, to apply color schemes, to centralize the ontology view and to reload the ontologies from the BRMS.

Rules and Queries

In the previous part of this section, the data model and UI of the demonstrator have been described. This subsection explains what kind of rules and queries have been implemented to achieve the demonstrator's functionality. In total, there are 63 rules and 28 different queries used in this demonstrator. As we omitted the figures in the prior subsection for an improved flow of reading, we relocated this subsection's listings to the annex as well (*cf.* section B.2).
First of all, we describe basic and auxiliary rules that have been implemented to support the more sophisticated ones.
The majority of the implemented rules add additional, transitive and dynamically calculated relations between classes. One of these rules in ObjectLogic is shown in Listing B.1. It states, that an *AnalysisResult a* is based on an *Analysis f* if *a* owns an *AnalysisAbility d* and *d* is created in a specific *f*. This statement can also be expressed as a property chain for easier readability: $owns(a,d) \wedge createdIn(d,f) \rightarrow basedOn(a,f)$.
This kind of rules reduces the complexity of the more sophisticated rules required by the various views. In Figure 7.17 on page 250 all the inferred relations have been highlighted.
In Listing B.2 the rule-based relation `uses_AnalysisResult` is introduced. Because `Assessment` is a subclass of `ProcessInformation`, it has a relation to `ProductInformation`. But not for all scenarios all the connected `ProductInformation` are necessary, but only a subset of them, i.e., the `AnalysisResults`. So this rule downsizes the solution set to the required instances.
The following type of rules was introduced because the correct values for the KPIs are not always obtainable as explained in section 4.4.3. We introduced two different attributes for the estimated and the actual value. The rules in Listing B.3 decide which one to use for the `Ability` property *cost*: if the actual cost is −1.0, the estimated value is used. Otherwise, the actual cost is applied.
Now, we want to clarify how the various rules and queries that are called and executed by the demonstrator are implemented.
The rule in Listing B.4 accumulates all the necessary properties for the query (*cf.* Listing B.5), which is called by the *planner matrix* web page. This kind of preparation for the queries has been done for all the other web page queries, too. This way, all the calculations, e.g., the calculation of the total time, are separated from the queries and the user role that is responsible for the interface between IT ap-

plication and business knowledge only needs to implement a query that calls the prepared rule.
The *planner matrix* rule is mainly based on two other rules.
Helper_Planner_Level1Properties_AnalysisResult is a class that is based on another rule to accumulate all the instance paths between `Level-1-Properties` and `AnalysisResults`. This object-creating rule was introduced to reduce the complexity of the rule and enhance the readability. Another advantage of this approach is the reduced real-time computation time that is needed when executing this rule by materializing sub-results.
The call for *f_helper_Planner_Matrix_transPI* executes the rules in Listing B.6 and Listing B.7. These rules calculate and return all the paths, including the `Ability` properties, between an `AnalysisResult` and the connected `ProductInformation`. Because this `ProductInformation` can be another `AnalysisResult`, that would be connected to another `ProductInformation` again, this rule has been designed as a recursive one. The termination condition is that `ProductInformation` is a `SolutionConcept`.
Besides gathering the KPIs for the different `Abilities` that are connected to the `ProcessInformation`, these rules also collect all the `Analyses` that are required for the corresponding path and are shown in the planner matrix view.
Additionally, this rule checks if a `ModelCreation` is executable via the rule-based property `executed_in` that is explained in Listing B.8.
`ModelCreations` are connected to 1-n `SolutionConcepts`. These `SolutionConcepts` are available at 1-m `Milestones`. The relation `executed_at` that connects `ModelCreations` to `Milestones` returns all the `Milestones` where the following statement holds true (*cf.* Listing B.8):
a model creation is only feasible, when all the connected solution concepts are available. Solution concepts can be available at different milestones. This means, that an initial solution concept is available at the first milestone. At the following milestones, the solution concept is updated to a newer version. Thus, the earliest time, when a model creation can be processed, is the minimum `StartDate` of all connected milestones. The last time the model creation can be processed is the maximum of the connected `StartDates`. The rule uses built-in aggregates to compute the minimum and maximum of the start dates and also returns the intermediate milestones.
In addition to the rules that are necessary to display the *planner matrix*, the web page makes use of less complex rules and queries that individually return all

available `Level-1-Properties`, all available `Level-2-Properties` and a complete list of the `Milestones` for the time line at the top of the *planner matrix*.
The *information chain* and *quality path* pages display a graph of concepts and their relations to each other. All the queries and rules that display the graphs are based on the rule in Listing B.9.
This rule gathers all possible paths from an `AnalysisResult` to a `ProductInformation` and stores the paths in the object *Helper_Informationchain*. Since the ending of these paths, the `ProductInformation`, can be either a `SolutionConcept` or another `AnalysisResult`, the created objects can be composed to create longer paths. The rules that are directly responsible for displaying the *information chain* and *quality paths* therefore only call this helper rule x times and add extra information, like the abilities for the quality paths.
What can be observed in this rule is, again, the heavy use of object-creating rules to improve readability and reduce complexity.
The *system analysis* page displays basic information that is stored in the ontology. For example, the `ProductInformation` that are not managed by a `Datasource`, e.g., a database, are returned by the query in Listing B.10.
The variable *?Datasource* is unified with the constant term *"DATENQUELLE_Not-Available"*. If the value is something else, the `ProductInformation` is managed. The other queries and rules for the *system analysis* that return information about the process information, the tools etc., are implemented analogously and thus are not explicitly explained here.
The other pages of the demonstrator use similar rules to the ones depicted on the previous pages resp. the annex. For example, the *property spider* queries lists for the properties and milestones. Additionally, the images that are displayed in the left panel are queried from the ontology.
The table with the *product information* queries all `SolutionConcepts` including their connected properties and some related classes, i.e., `Milestone`, `Datasource` and `Level-2-Property`. For the *free ontology browsing*, the complete ontology, including all entities, is queried and then displayed.

7.5.5. Evaluation

Our evaluation is based on the business requirements regarding the prototype and the methodology for our SWT-based approach. Therefore, we review our

raised requirements, gather and analyze user experiences and describe insights made during the various methodology phases.

Evaluation Business Requirements

The primary goal of this case study, resp. the business model, was to cover the relationships between process, product, methodical and IT knowledge, as presented in the method model in chapter 4, and hence enable end users to analyze methods with the defined comparison criteria within this orchestration. The PDD use case, the business scenario behind this demonstrator, evidently covered these various aspects.
Most of the core method ontology's concepts from section 4.3 can be rediscovered in the PDD meta model, presented in Figure 7.17. Albeit, this use case model is a special case with particular requirements, of course, and thus does not cover every concept exactly.
`Methods` are applied in `Processes`, which in turn are supported by `Tools`. We also distinguish between concrete and abstract methods in the PDD meta model, by employing the class `Ability`. This class acts like the `Concrete Methods` in our core method ontology by relating method utilizations with a quality attribute, like the time, maturity, cost or general quality. Thus, it depends on a concrete application in the PDP, combined with a specific loading case. The class `Method`, on the other hand, does not directly relate to the defined KPIs and represents the `Abstract Method`. The `ProductInformation` that are necessary to conduct a process step like an analysis, assessment or model creation, resp. the method behind, fit in the class `Resource` in our method model, although a `Resource` is intended to be more coarse-grained and cover more kinds of resources. Besides, a `ProductInformation`'s availability, and hence a consequent method application, is expressed by a related `Milestone` of the PDP. This circumstance is implemented more specific than in our introduced method model.
Furthermore, for the provided use case, we did not need to understand an applied method's internal mechanics in particular. Therefore, we refrained from the introduction of the class `Procedure` in the use case model. Nevertheless, additional information for each class can and have partially been deposited by making use of the `Document` and `Image` classes.

Besides, we did not model a business layer that covers enterprise goals, like in an EA meta model. Nevertheless, a methods' `Goal` in our use case's context is obvious: to attain a baseline assessment, due to the model creation and analysis, of a given product property.
The resulting ontologies, together with the rules and queries, allowed us to define complex comparison criteria for CAx methods. Thus, the raised business questions in the introduction of this case study can be answered to our satisfaction, entirely. Method chains, computed by rules and visualized in the application, enable the end users to understand its coherences and dependencies, which is beneficial for method engineers as well as project leaders. Other end user roles benefit from the sundry other views for monitoring, comparing and analyzing the use case. For instance, IT specialists or enterprise architects can analyze the dependencies between `Data Sources` or `Tools` and the processes and methods they are used in.

End user interviews

Besides comparing our previously gathered requirements with the implemented results and our approach, we also performed individual end user interviews. Therefore, we introduced the case study, roughly explained the underlying SWTs where needed, presented the prototype in general and then emphasized on the anticipated interesting aspects for the respective interviewee.
Subsequently, we conducted a semi-structured, but mostly qualitative, interview, which means that we had prepared and then completed a questionnaire, followed by an open discussion afterwards.
The goal of these interviews, on the one hand, was to gather general opinions and feedback about our approach, the underlying business scenario and the developed prototype and its usability by putting it to the test. Therefore, our intention was to compare the currently used workflows with our scenario. On the other hand, we wanted to learn about emerging opportunities, i.e., possible future expansions and a resulting outlook.
The interviewees covered a wide range of possible end user roles: persons on a managing level that are responsible for the overall development, people responsible for the establishment, monitoring and adherence to product properties, DEs that work with specific car sections, technical project managers, enterprise archi-

tects, employees that are responsible for a particular car project, CAx engineers and process managers in virtual and physical R&D departments alike.
The questionnaire has been designed jointly by a DE and me. Thus, it consisted of various issues, covering particular business and engineering related questions, that have been asked independently of my thesis and hence, are irrelevant and not itemized here. Furthermore, the exact results are company-internal and therefore, the outcomes are described in a much more abstract way.
Nevertheless, the relevant sections covered questions about

- the process in general,
- the required level of detail for each user role and usability-related questions,
- the currently used work flows for the described scenario,
- questions about the currently practiced and preferred KA process
- and the relevance of a link between an ontological entity and its source document.

The answers and statements about the topics have been widespread and different, as expected, among the interviewed user roles.
In general, the end users conceived and welcomed the possibilities opening up by such a system and approach, because it improves the understanding about the depicted concepts' coherences and dependencies. The more often than not positive feedback was generally accompanied with proposals about additional and desired functionalities.
In order to analyze the demonstrator's usability we cooperated with IBM. They provided a guideline in order to test and evaluate the UI (Bonis and Bellino 2011), which included suggestions to select appropriate users, prepare the testing tasks and general hints to set up the testing environment. This way, the end user's feedback helped us to identify the UI elements in need of improvement.
Above all, DEs and other roles, working low-level in the PDP, required more detailed information about their field of activity, which can be very distinct. Besides, the people working with particular components or assemblies desired filtered views for exactly their domain. Additionally, specific processes in the vehicle domain, e.g., facelift, platform or derivative development, partially apply their very own methods to test the respective behavior and properties. Thus, sharing this knowledge with other domains would be unreasonable, according to the interviewee. Furthermore, we were able to confirm that methods are seldom applicable for the car development in general but have to be specified for a single car project.

Managing roles, on the other hand, were naturally more interested in high-level views, but were happy to have the option to dig into details, when needed; especially, when planned processes do not operate as expected, e.g., if deadlines are threatened to lapse or other errors occur. Precisely through the presented approach, they have primarily been able to detect and visualize the belonging processes in this particular manner.
Regarding the currently practiced KA and operating methodologies, the end user interviews covered our expectations from the identified requirements. Lots of the applied knowledge is not formally written down, i.e., it existed only in the heads and drawers of the interviewees. Nevertheless, several departments make use of application-specific project managing, requirements and of course PLM or PDM tools. Other units utilize structured data in the form of spread sheets.
Some of these systems even offer links to source and additional documents, like legal documents, regulations or brand instructions, for the presented information. Therefore, featuring such an approach in our prototype has been very well received.
However, certain criticisms and indications were expressed, too. As mentioned, the majority would benefit from the use of such an approach, the resulting KB and application. As expected, the problem is acquiring, updating and validating this knowledge, though. Employees, working to capacity, cannot maintain the KB manually, in addition to their daily work. In addition, they usually need training in order to use the modeling tools correctly and efficiently.
Furthermore, the roles providing the required knowledge are not congruent to the ones benefiting from such an application and approach. Principally, the derived and abstract knowledge that emerges from the underlying low-level information is most useful for certain roles. However, these role groups only consume and do not provide the process and method information. Besides, having these distinct role groups and knowing about the maintenance challenges encourages a mistrust in such KBs, because the recipient cannot simply confirm its correctness and completeness, according to the interviewees.

Evaluation Methodology

In this subsection, we present our experiences and insights with the development and prototypical usage of the application which is based on our methodology,

viz., the KA, modeling/authoring of the TBox and rules and queries, the populating/formalization of data and its the execution in an SWT-based application. The main technical objective was to separate business knowledge, domain knowledge and IT code by applying SWTs and consequently to focus the required work of the involved roles to their respective scope. Furthermore, we evaluated the SWTs in conjunction with our domain of methods and EA elements.

During the KA, I, as a KE, worked tightly together with DEs to come up with a domain model and the business knowledge for our demonstrator. They provided the inside knowledge required for our ontologies, the terminology and the semantics of our BRs on different channels: Our primary source for this knowledge was the outcome of discussions with DEs. Furthermore, we extracted structural information from spreadsheet files and database schemata and created ontologies thereof. The model's entities could then be labeled according to these sources. Different ontology and rule modeling tools, e.g., Protégé and Ontostudio, have been used in this process. Complex and more sophisticated statements had been realized by the KEs, in consultation with the DEs. Therefore, the graphical model and rule representations were most helpful, because it became apparent that it is easier to grasp the semantics if it is displayed in a graphical version to the DEs.

The resulting model's TBox and rules/queries correctness and completeness was validated in different ways. On the one hand, we used the methodology of expert-aided modeling in order to check the integrity of our ontologies (Hoppenbrouwers, Nijssen, and Van Leeuwen 2012). On the other hand, we implemented a testing environment for our business and domain knowledge, as seen in section 7.5.3, i.e., a test ontology and a test rule/query set. This way, general mistakes made while modeling, for instance, missing relations or wrong cardinalities, could be detected and corrected. Besides, we implemented rules and queries for data quality assessment as user constraints as indicated in section 6.4 about method artifacts in order to identify and detect, for instance, orphaned, missing and duplicated entries.

For the populating of our models, we had many spreadsheets containing data for our ABox and we could even connect data base systems where the knowledge was already formalized.

The application of NLP methods has already been evaluated in section 7.2 where we presented a use case that links ontologies and written policies. For the anticipated scope of the PDD use case, the whole process has not been feasible, because the domain has not been written down in documents of a form that these NLP

methods could be applied to. Our goal was to model the meta level of methods and the PDD and not to describe methods in such detail as described in detailed policies and regulations about particular method applications and tests. Therefore, we did not automatically extract model data or meta model elements from documents written in natural language. However, our model provides ways to link concepts to their source or other `Documents` that offer hints or guidelines according to the method description elucidated in section 4.6.2.
The architecture and technical foundation of our SWT-based application proved effective with some minor reservations. First of all, the entire information, i.e., the artifacts, that is visible in the prototype's UI, like the duration or total cost of method chains, the catalogs and matrices are calculated and inferred by the rules, ontologies and queries in the semantic middleware, which has been our primary technical requirement. The rules and queries use the domain knowledge available in the ontologies which simplifies, explains and clarifies their meaning and results enormously. An IT supplier implemented the application code that just displayed the results of our queries. As a consequence, we as DEs and KEs could model and maintain our business and domain logic/knowledge independently. Therefore, the separation of responsibilities and tasks has been achieved completely.
A minor drawback we encountered was the slow or even impossible processing of the huge amount of data we tried to compute, which did not emerge when using only a testing sample of our data. The supported method chains and alliances could get very long and large, which increased our solution space massively due to the combinatorial complexity. However, by materializing the most common and intensive rules and queries, i.e., doing the time-consuming calculations offline and thus making these complex constructs available in near real-time for the querying application, this drawback could be partially compensated. We did not solve this issue completely, however, data and rule/query optimization and the use of high-performance triple stores and hardware can certainly reduce this effect. This topic has not been a primary concern, though.

7.5.6. Conclusion

The case study's prototypical application has been developed in order to support DEs and managers by monitoring and planning the PDD in the car innovation

and R&D cycles. In general, a challenge for the PDPs implemented in enterprises is to better integrate virtual development and engineering disciplines, like the application of virtual methods instead of physical ones, in order to cope with a higher product diversity and shortened development cycles that cannot be solved and covered by conventional, physical methods alone anymore. This case study exemplifies how methods, product information, properties and processes can be modeled in the PDD by applying the methodology described in chapter 5. The prototype allows involved roles to determine the most suitable methods and method chains, along with the appropriate tools, in order conduct a process action that models, analyses or assesses a particular product property.
By separating software code, domain knowledge and business logic, attributable to the use of SWTs, we improve the synchronization and individual maintenance of software, business and domain knowledge life cycles during the development. Furthermore, due to the SWTs openness, new knowledge can be mapped and integrated by design. The technology and developed prototype including its architecture allow us to infer and calculate complex statements. The results, however, heavily depend on the quality and quantity of the underlying data at the bottom level of the modeled method chains. Conveniently, the resource data can be augmented with product information represented in BOMs, like the ontologies and rules presented in section 7.3. Besides, we showcased how such KBs can be augmented in general with our developed method ontology in chapter 4. During this case study, we used another, previous version ontology in order to represent our methods. However, both meta models match and therefore it can be extended as elucidated.
Figure 7.21 illustrates that all the main sections of this thesis have been applied and evaluated in this chapter. Our model incorporates EA elements, like tools, data sources and process information, and we showcased various implemented and evaluated artifacts concerning methods and their contexts. These artifacts make use of method metrics, like time consumption, method maturity and costs in order to support users in their method selection. Furthermore, methods are directly linked to process actions they are used in, in combination with the resources required to conduct them which allows the computation of method chains and alliances which in turn infers the tool chains required to conduct the processes. The developed views representing the different artifacts have been evaluated together with various DEs, i.e., possible user roles, which enabled us to match and sharpen their profiles concerning the described domain. One role usu-

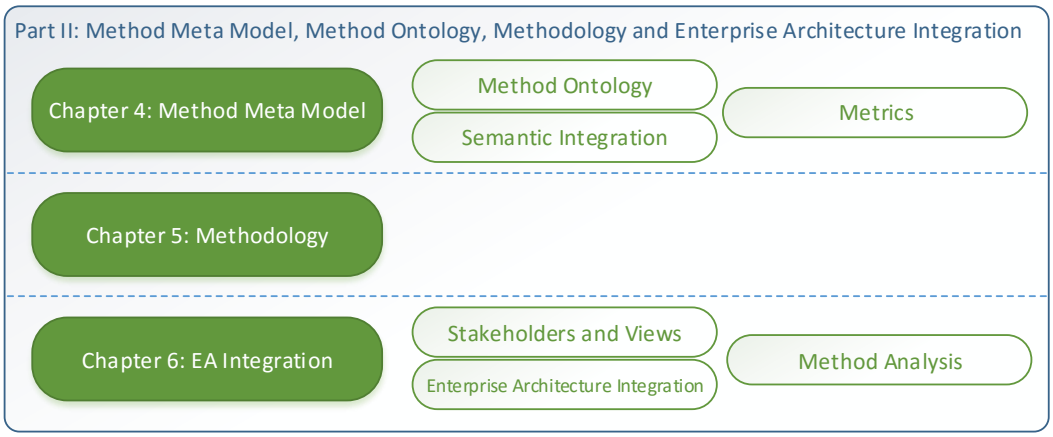


Figure 7.21.: The PDD case study applied every major section of this thesis.

ally works in a very specialized domain and is not interested in the entirety of the provided classes and entities. Therefore, the queries and concerning views can be partitioned into even smaller excerpts. These outcomes have been considered in chapter 6.

The biggest challenge during the development was to collect all the necessary knowledge to model the ontology TBox and the gathering of the facts to populate it, i.e., the methodology's KA phase. Nevertheless, many iterations over the rule modeling has led to a performant demonstration system that has been realized as a web application. Furthermore, in addition to the features and methodology developed so far, it is necessary to provide further guidelines, architecture and design recommendations for the development of this type of KM systems based on SWTs.

"Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest."

Isaac Asimov (1920 – 1992)

8

Conclusion

8.1. Contributions

Newly emerging and changing markets and their diverse demands lead to an increased product variability, increasing functional complexity and regulatory requirements in the product development. Simultaneously, however, costs and development times shall be reduced and qualities increased which causes conflicting goals. One contribution to cope with these challenges is the introduction and promotion of virtual product development which promises to overcome these defiances by substituting conventional, physical tasks with virtual ones. Thereby, the considered tasks can usually be repeated more often and performed faster at lower costs. Besides, Knowledge Management (KM) is simplified which increases process flexibility, because the regarded resources exist as data that can be managed and reused, supported by Information Technology (IT) solutions, for instance, Computer Aided *x* (CAx) technologies. These tasks in a process are usually supported by numerous methods, for instance, design methods, which in turn are conducted by utilizing tools which implicates that one goal is to supersede conventional methods and the connected tools with virtual or digital ones. The challenges of implementing such a virtual product development are manifold, though, as exemplified in this dissertation's first part.

The illustration in Figure 8.1, also know from this thesis' introduction, depicts the types and characters of knowledge the product developed is concerned with.

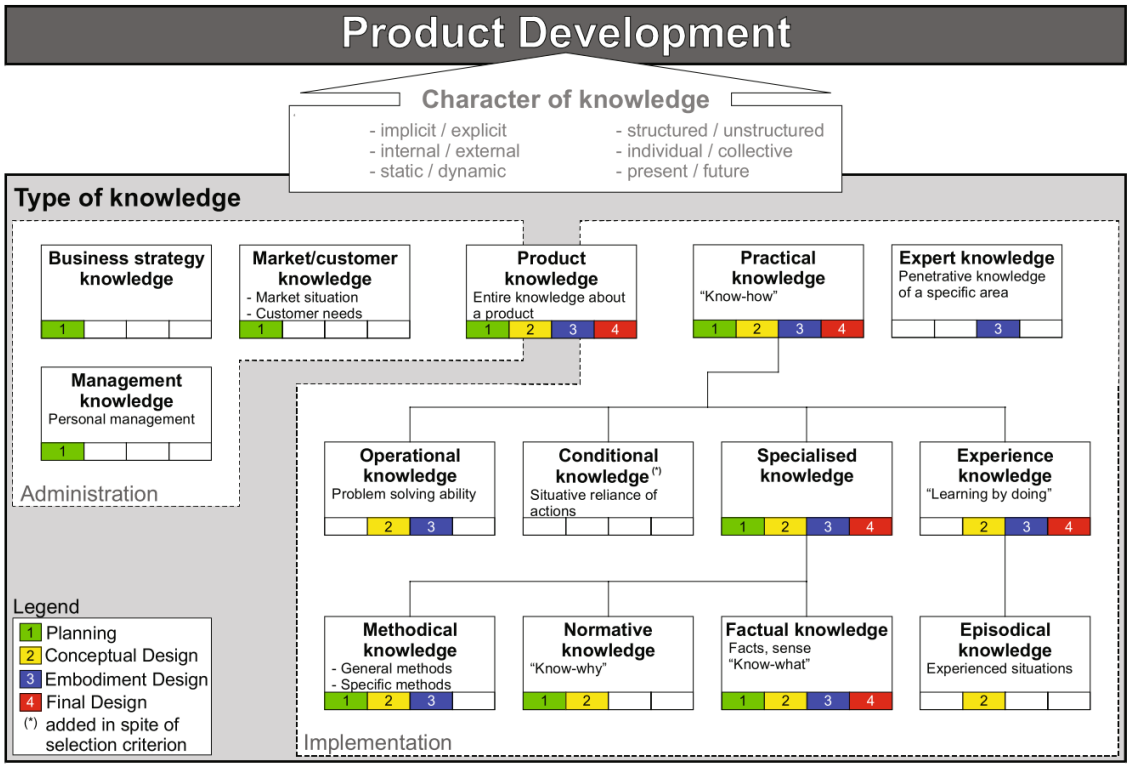


Figure 8.1.: A theoretical model of the structure of knowledge in the product development (Roth, Binz, and Watty 2010).

In the course of this dissertation, we have shown how domain and business knowledge and logic in the product development, especially method and Enterprise Architecture (EA) knowledge, can be gathered, formalized and integrated using Semantic Web Technologies (SWTs) which leads to more transparency and awareness and hence enables analyses and the assessment of concerted knowledge in order to support decision-making in an enterprise. Thereby, virtual methods can be promoted. With reference to Figure 8.1, the EA knowledge fits into both categories of knowledge, i.e., administration and implementation knowledge, exemplified as business strategy, management knowledge and practical knowledge types, while our method ontology, combined with further information, such as product knowledge and metrics, extended by rules and queries, fits into the implementation category of knowledge. Our considered concepts can also be partitioned into other architectures, such as organizational and operational layers as elaborated in chapter 4 or a business architecture and subsequent architectures in Enterprise Architecture Management (EAM), though, be-

cause "there is no consistent taxonomy of knowledge due to its multidisciplinary character" (Roth, Binz, and Watty 2010). As a consequence, we mainly differentiate between domain knowledge, business knowledge and business logic.
The developed ontologies presented in this thesis depict the domain of design methods, including their context information, for instance, references to the processes they are used in, quality attributes, the methods' in- and outputs and the tools that support those methods. The defined method meta model is based on industrial insights as well as an elaborated literature research. As a consequence, it is on an abstract level and implemented as an upper ontology in order to be able to harmonize numerous method classifications from diverse backgrounds, for instance, from the CAx domain, as exemplified during the case studies. The integration of additional domain knowledge, like product knowledge or quality attributes, using SWTs in the form of Domain ontologies (DOs) and queries and rules for their matching has been showcased, as well. Besides, we displayed the integration and matching of the foundational resource data, exemplified in the Bill of Material (BOM) case study, using ontologies and rules, Frame Logic (F-Logic) in particular. Furthermore, we have demonstrated how ontologies can be used to support different vocabularies for numerous stakeholders, scopes and point of views by utilizing the standard Simple Knowledge Organization System (SKOS) to annotate Web Ontology Language (OWL) ontologies with alternative labels. This has been showcased in our evaluation about lexicalized ontologies in section 7.2 which, next to providing the diverse vocabulary, supports the annotation of concepts and entities with texts from regulations or any other documents. Besides, the developed Property-Driven Development (PDD) demonstrator showcases how using multilingual labels and descriptions in ontologies realize an application that can be used by English- and German-speaking users.
Taken together, we have fulfilled *Objective 1*, the integration of method knowledge including the methods' contexts.
According to *Objective 3*, we implemented and integrated an EA ontology with our method ontology and elucidated how further EA extensions can be matched, which bears many mutual benefits. Thereby, we can exploit relationships and dependencies between methods, processes, such as a Product Development Process (PDP), IT, business objects and the enterprise strategy which increases their flexibility, quality and efficiency. For example, it allows predicating even more complex statements on expected qualities, costs, time consumption or other metrics and ensuing Key Performance Indicators (KPIs) which is a strong motive for Do-

main Experts (DEs) when comparing and hence selecting an appropriate method or even method-mixes best suited for the task at hand in the product development. Various metrics have been introduced in the metrics ontology and a subset has been evaluated in the case studies. Besides, mandatory method applications, for instance, due to given functional and regulatory requirements, can be considered by stating this circumstance using rules. In addition, it enables stakeholders, that have been defined in this dissertation, to conduct further novel kinds of analyses, like the analysis of relationships on a strategic level, promoting a targeted method development and shutdown; on the business level, e.g., between roles, processes and methods; for rising data quality by finding blank spots, lacunae or duplicates; and also on a business to IT level, for instance, by providing an overview of the applied method software tools. For example, these insights can be used to promote an orchestration of the enterprises CAx tools, methods and systems. Correspondingly, we have defined views, method architecture artifacts and business goals that are supported by these novel analyses. Thereby, we have fulfilled *Objective 4*.

The methodology for realizing our approach has been explained, conducted and evaluated throughout this dissertation in compliance with *Objective 2*. Using SWTs for this approach is very valuable, because the modeled knowledge can be verified, new knowledge can be inferred and external knowledge, like public Semantic Web (SW) ontologies, can be harnessed in the enterprise scenario. However, the primary reason has been the separation of domain knowledge, business knowledge and logic in order to support different lifecycles, scopes and knowledge owners which is beneficial for diversely evolving knowledge and variable innovation and maintenance cycles.

The most considered methodology phase has been the Knowledge Acquisition (KA), i.e., gathering and formalizing the various types and characters of knowledge that are distributed throughout an enterprise by applying methods like Natural Language Processing (NLP) and expert-aided modeling. Having means to formalize this knowledge into a correct and reusable standard is obviously a key factor for a successful application of the succeeding phases of authoring and execution. However, the application, documentation, the execution and the maintenance of the formalized method management and its context is expensive, can be complex and is wedded to effort, because it is still mainly a manual procedure. During our evaluation task, though, we learned that the knowledge used in our anticipated method expert system only partially changes. That means the main-

tenance is less time-consuming than gathering the knowledge in the first place and reusing the gathered knowledge in coming projects will reduce time- and cost-consuming KA phases and has the benefit that the knowledge can be maintained by the responsible roles.
The new concepts and contributions, namely the use of SWTs for integration of our method ontology and EA, along with the extending ontologies and the corresponding queries and rules, have been modeled, formalized and executed with a chosen set of test scenarios, for instance, the development of the demonstrators in the case studies, that showcase the methodology. In fulfillment of *Objective 5*, the scientific publications and presented demonstrators, that have been evaluated in a real industrial environment with numerous DE, showcase the feasibility of our approach and a business solution that helps integrating the physical and virtual development.
Summarized, we have successfully fulfilled all the developed objectives from our introduction and thereby made contributions to the according disciplines, such as KM, (virtual) methods in product development, EA and the integration, e.g., ontology and rule mappings and transformations, and evaluation of SWTs for the modeling and analysis of this knowledge in an industrial setting.

8.2. Outlook

The utilization of SWTs and the introduced methodology for the modeling and following analysis and assessment of method knowledge, combined with EA concepts and further contexts, bears many auspicious possibilities, but also some cruxes, when shifting from conventional to virtual product development.
The acquisition and management of business and domain knowledge in formalized ontologies, rules and queries improve their interchange, interconnection, synchronization, sharing, reusability, re-deployment, up-to-dateness, the integration and allows maintaining this knowledge by the appropriate roles in the enterprise, because (conceptual) domain knowledge, (operational) business rules and an implemented data model have different lifecycles, scopes and owners. Thus, the use of SWTs increase the flexibility, quality and efficiency of the development process, allowing enterprises to meet the increasing market demand of product diversification, increasing functional complexity and regulatory requirements, for instance, due to an improved integration of virtual development. The

demonstrated approach allows monitoring and planning the method landscape strategically and, for a particular process action, determining the most suitable method-mix. Future improvements in the form of new integrations, also from completely new domains, and analyses lead to opportunities like the conduction of method impact analyses or gap analyses. By comparing a baseline with a target architecture, new aimed method development projects in a method portfolio management can be initiated to rise the company's effectiveness, efficiency and overall quality. In general, such a KM approach leads to more awareness and transparency which allows finding competencies in the enterprise, avoiding duplicate or other undesirable developments, transferring and exchanging of tools, data and knowledge between projects or taking into account legal requirements and regulations during the development. Additionally, individualized views will also lead to a shorter training periods and a higher usability for users. By formalizing the knowledge and logic into accepted standards like Semantics of Business Vocabulary and Business Rules (SBVR), OWL, SPARQL Protocol And RDF Query Language (SPARQL) and Rule Interchange Format (RIF) the enterprise has the freedom to exchange IT systems and thus is not bound to specific vendors. Emerging standards, like Decision Model And Notation (DMN) (Object Management Group 2014) which is supposed to integrate business process knowledge, like Business Process Model and Notation (BPMN) models, together with rules (RIF) and vocabularies (SBVR) or the standard Semantic Information Modeling for Federation (SIMF) (Casanave 2012) will further improve the integration of diverse business and domain knowledge and logic. The considered standards are supported by standardization groups, like World Wide Web Consortium (W3C) or Object Management Group (OMG) and thus guarantee a reusability and interchangeability. Adding new concepts and writing new rules to add more facets of the development process does not automatically entail reimplementing the already in place software solutions that make use of the formalized knowledge. Expanding the ontological schema or merging and matching ontologies with already existing internal or external ones is well supported by appropriate tools and allows enhancing the Knowledge Base (KB) easily. The responsible DEs and Knowledge Engineers (KEs) can manage the vocabulary and adapt rules to more complex scenarios and dynamic business parameters on their own without the need to modify the IT application itself and thus can experience the effects immediately. This will help in the future, when individualized and customized applications and views to support the development have been implemented in an IT

landscape that is based on services that provide the relevant information for an individual. This provided knowledge, however, can be extracted from different ontologies and rule sets distributed in the IT architecture.
Besides, other domains, such as Digital Prototypes (DPs) or the digital factory can also benefit from the presented approach. This digital factory is paradigm for the integration of processes, digital models, methods and tools for the purpose of planning, evaluating, improving and safeguarding production systems, i.e., the shift from a conventional factory to a digital one.
Despite the manifold auspicious possibilities, some cruxes and challenges still have to be mastered. First of all, topics like proof, trust and security have to be considered. They are already part of the SW Stack, but mature, meaningful and robust concepts yet have to be developed. Certainly, data validation and verification have already been considered, partially inherently due to the used technologies. Furthermore, user constraints can be implemented using rules and queries which improves the data quality. However, such data quality and KM processes in general have to be implemented in the enterprise and be made transparent in order to provide correct and meaningful knowledge. Otherwise, end users, like DEs, will doubt the proposed solutions, because they do not trust the KB's correctness and completeness. The developed demonstrators in this thesis offer views to dig deep into a detailed visualization of the proposed model, like method chains, though, so the end user can confirm their correctness autonomously. However, this process is in need of improvement, because it is too time-consuming. Furthermore, offering different views to the diverse user roles also requires introducing a rights- and role management.
Another aspect that can be improved is the KA and model authoring. Knowledge models need to be discussed, modified, agreed upon and released by sundry roles and the once acquired knowledge has to be kept up-to-date. This is still mainly a manual and hence expensive task today. Technologies, like NLP tools are often too complex to be used by laymen or not matured enough yet to capture and integrate knowledge automatically from existing sources. We know from experience, that solutions that would require DEs to learn new modeling languages or complicated workflows in order to formalize their knowledge are rarely accepted and therefore do not prevail. As a consequence, next to the improvement of some techniques and tools of the methodology, business processes and workflows have to be adjusted in order to report and integrate newly generated knowledge in a usable form.

Part IV.

Annex

Printed Sources

Aier, Stephan, Christian Riege, and Robert Winter (2008). "Enterprise Architecture - Literature Overview and Current Practices". In: *Wirtschaftsinformatik* 50 (4), pp. 292–304. ISSN: 0937-6429.

Albers, A., N. Reiß, N. Bursac, L. Schwarz, and R. Lüdcke (2015). "Modelling Technique for Knowledge Management, Process Management and Method Application - a Formula Student Exploratory Study". In: *Modelling and Management of Engineering Processes*. Springer Berlin Heidelberg, pp. 151–162. ISBN: 978-3-662-44008-7.

Albers, A., N. Reiß, N. Bursac, J. Urbanec, and R. Lüdcke (2014). "Situation-appropriate method selection in product development process - empirical study of method application". In: *NordDesign 2014*. Espoo, Finland, pp. 550–559.

Angele, Jürgen, Michael Kifer, and Georg Lausen (2009). "Ontologies in F-Logic". In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. Springer-Verlag Berlin Heidelberg, pp. 45–70.

Arara, Ahmed and Djamal Benslimane (2004). "Towards Formal Ontologies Requirements with Multiple Perspectives". In: *6th International Conference, FQAS 2004*. Vol. 3055. Lyon: Springer Berlin Heidelberg, pp. 150–160. DOI: 10.1007/978-3-540-25957-2_13.

ASAM (2006). *ASAM Solutions Guide 2006*.

– (2013). *ASAM Solutions Guide 2013*.

Auer, Gerhard et al. (2011). *Enterprise Architecture Management – neue Disziplin für die ganzheitliche Unternehmensentwicklung*.

Baader, Franz, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F Patel-Schneider, eds. (2003). *Description Logic Handbook: Theory , implementation , and applications*. New York, NY, USA: Cambridge University Press. Chap. 2, p. 622. ISBN: 0521781760. DOI: 10.2277/0521781760.

Baader, Franz, Ian Horrocks, and Ulrike Sattler (2007). "Description Logics". In: *Handbook of Knowledge Representation*. Ed. by Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. Elsevier.

Baader, Franz and Ulrike Sattler (2001). "An Overview of Tableau Algorithms for Description Logics". In: *Studia Logica*. Lecture Notes in Artificial Intelli-

gence 69 (1). Ed. by R Dyckhoff, pp. 5–40. ISSN: 00393215. DOI: 10.1023/A:1013882326814.

Badke-Schaub, P., J. Daalhuizen, and N. Roozenburg (2011). "Towards a Designer-Centred Methodology: Descriptive Considerations and Prescriptive Reflections". In: *The future of design methodology*. Ed. by Herbert Birkhofer. Springer London Limited. Chap. 16, pp. 191–197.

Bao, Jie (2007). "Representing and reasoning with modular ontologies". PhD thesis. Iowa State University.

Bartz, Rainer (2009). *ASAM ODS Open Data Services Specification Version 5.2.0*.

Barzdins, Janis, Guntis Barzdins, Karlis Cerans, Renars Liepins, and Arturs Sprogis (2010). "UML Style Graphical Notation and Editor for OWL 2". In: *Perspectives in Business Informatics Research*. Springer Berlin Heidelberg, pp. 102–114. DOI: 10.1007/978-3-642-16101-8_9.

Becker, J. and G. Vossen (1996). "Geschäftsprozeßmodellierung und Workflow-Management: Eine Einführung". In: *Geschäftsprozeßmodellierung und Workflow-Management — Modelle, Methoden, Werkzeuge*. Ed. by G. Vossen and J. Becker. Bonn: International Thomson Publishing, pp. 17–26.

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). "The Semantic Web". In: *Scientific American*. Lecture Notes in Computer Science 284 (5). Ed. by Asunción Gómez-Pérez, Yong Yu, and Ying Ding, pp. 34–43. ISSN: 00368733. DOI: 10.1038/scientificamerican0501-34.

Berrueta, Diego, Roman Korf, Eva Maria Kiss, Jeroen Hoppenbrouwers, Sjir Nijssen, Adeline Nazarenko, Adil El Ghali, Hugues Citeau, and Christian de Sainte Marie (2011). *D6.1 - Specification of the ONTORULE platform*. ONTORULE Project.

Binz, H., A. Keller, M. Kratzer, M. Messerle, and D. Roth (2011). "Increasing Effectiveness and Efficiency of Product Development - A Challenge for Design Methodologies and Knowledge Management". In: *The future of design methodology*. Ed. by Herbert Birkhofer. Springer London Limited. Chap. 7, pp. 79–90.

Birkhofer, Herbert, ed. (2011). *The future of design methodology*. Springer London Limited. ISBN: 9780857296146.

Birkhofer, Herbert, H. Kloberdanz, B. Berger, and T. Sauer (2002). "Cleaning up Design Methods - Describing Methods Completely and Standardised". In: *Proceedings of DESIGN 2002, the 7th International Design Conference* 30, pp. 17–22.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). "Freebase: a collaboratively created graph database for structuring hu-

man knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250. (Visited on 06/16/2014).

Bonis, Sophie de and Catherine Bellino (2011). *D2.5 - Final usability report : evaluation and conclusions*. ONTORULE Project.

Borgida, Alex and Luciano Serafini (2003). "Distributed Description Logics: Assimilating Information from Peer Sources". In: *Journal on Data Semantics* 1, pp. 153–184.

Bouquet, Paolo, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt (2003). "C-OWL : Contextualizing Ontologies". In: *Second International Semantic Web Conference ISWC03* 2870, pp. 164–179.

Brachman, Ronald and Hector Levesque (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann Series in Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., p. 381. ISBN: 1558609326. DOI: 10.1146/annurev.cs.01.060186.001351.

Brambilla, Marco, Jordi Cabot, and Manuel Wimmer (2012). *Model-Driven Software Engineering in Practice*. Vol. 1. Morgan & Claypool Publishers. ISBN: 9781608458820. DOI: 10.2200/S00441ED1V01Y201208SWE001.

Brambilla, Marco, Stefano Ceri, Federico Michele Facca, Irene Celino, Dario Cerizza, and Emanuele Della Valle (2001). "Model-Driven Design and Development of Semantic Web Service Applications". In: *ACM Transactions on Internet Technology (TOIT)* 8 (1).

Braun, Andreas, B. Ebel, and A. Albers (2013). "Activity-Based Modeling and Analysis of Product Engineering Processes". In: *Smart Product Engineering: Proceedings of the 23rd CIRP Design Conference*. Ed. by M. Abramovici and R. Stark.

Braun, Christian, Martin Hafner, and Felix Wortmann (2004). *Methodenkonstruktion als wissenschaftlicher Erkenntnisansatz*. St. Gallen: Institut für Wirtschaftsinformatik, Universität St. Gallen.

Braun, Thomas (2005). "Methodische Unterstützung der strategischen Produktplanung in einem mittelständisch geprägten Umfeld". PhD thesis. Munich, Germany: TU Munich.

Braun, Thomas and Udo Lindemann (2003). "Supporting the selection, adaptation and application of methods in product development". In: *International Conference on Engineering Design, ICED'03*. Design Society, p. 1.

– (2004). "Method adaption - a way to improve methodical product development". In: *International Design Conference - DESIGN 2004*. Dubrovnik, Croatia.

Breitling, Thomas, Theodor Großmann, and Albrecht Zöller (2009). "Digitale Prototypen unterstützen Entwicklung". In: *ATZextra* 14 (1), pp. 162–171.

Brinkkemper, Sjaak (1996). "Method engineering : engineering of information methods and tools". In: *Journal of Information & Software Technology* 38 (4), pp. 275–280.

Buchert, Tom, Alexander Kaluza, Friedrich a. Halstenberg, Kai Lindow, Haygazun Hayka, and Rainer Stark (2014). "Enabling product development engineers to select and combine methods for sustainable design". In: *CIRP Conference on Life Cycle Engineering*. Vol. 15. Trondheim, Norway, pp. 413–418. DOI: 10.1016/j.procir.2014.06.025.

Buckl, Sabine, Florian Matthes, A. Ernst, and J. Lankes (2008). *Enterprise Architecture Management Pattern Catalog*. Munich, Germany: Software Engineering for Business Information Systems (sebis).

Bulles, John, Jean-paul Koster, Sjir Nijssen, Stan Dieteren, Inge Lemmens, Florine Durand, Roman Korf, and Sylvain Dehors (2009). *D1.2 Transforms from SBVR to ontologies and rules*.

Calvanese, Diego, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi (2001). "Reasoning in expressive description logics". In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Amsterdam: Elsevier Science Publishers (North-Holland). Chap. 23, pp. 1581–1634.

Cameron, By Brian H and Eric Mcmillan (2013). "Analyzing the Current Trends in Enterprise Architecture Frameworks". In: *Journal of Enterprise Architecture* 9 (February), pp. 60–71.

Chomicki, Jan (2008). *Data Integration : Schema Mapping*. University of Buffalo.

Chroust, Gerhard (1992). *Modelle der Software-Entwicklung - Aufbau und Interpretation von Vorgehensmodellen*. Munich: Oldenbourg-Verlag.

Corcho, O., Mariano Fernández-López, Asunción Gómez-Pérez, and A. López-Cima (2005). "Building legal ontologies with METHONTOLOGY and WebODE". In: *Law and the Semantic Web*. Lecture Notes In Computer Science 3369. Ed. by P Casanovas, J Breuker, and A Gangemi, pp. 142–157. ISSN: 03029743.

Cronholm, Stefan and Pär Ågerfalk (1999). "On the Concept of Method in Information Systems Development". In: *22nd Information Systems Research In Scandinavia (IRIS 22)*. Ed. by T. Käkölä. Keuruu, Finland.

Davenport, Thomas H. (1993). *Process innovation: reengineering work through information technology*. Boston, MA, USA: Harvard Business School Press, p. 337. ISBN: 0875843662. DOI: 10.1016/0923-4748(94)90026-4.

– (1994). "Saving IT's Soul: Human-Centered Information Management." In: *Harvard Business Review* 72 (2), pp. 119–131. ISSN: 00178012.

De Bruijn, Jos, Philippe Bonnard, Hugues Citeau, Sylvain Dehors, Stijn Heymans, Roman Korf, Jörg Pührer, and Thomas Eiter (2009). "D3.1 - State-of-the-art survey of issues". In:

de Sainte Marie, Christian, Miguel Iglesias Escudero, and Peter Rosina (2011). "The ONTORULE Project : Where Ontology Meets Business Rules". In: *Web Reasoning and Rule Systems* 6902, pp. 24–29.

Decker, Stefan (2002). "Semantic Web Methods for Knowledge Management". PhD thesis. Karlsruhe, Germany: Universität Fridericiana zu Karlsruhe (TH).

Denning, Peter J. (1976). "Fault Tolerant Operating Systems". In: *ACM Computing Surveys* 8 (4), pp. 359–389. ISSN: 03600300. DOI: 10.1145/356678.356680.

Dietz, Jan L. G. (2006). *Enterprise Ontology: Theory and Methodology*. Enterprise Engineering Series. Springer Berlin Heidelberg, p. 248. ISBN: 9783540291695. DOI: 10.1007/3-540-33149-2.

Djellal, Asma, Mounir Hemam, and Zizette Boufaida (2010). "An Extension of the Ontology Web Language with Viewpoint and Fuzzy Notions". In: *International Symposium on Modelling and Implementation of Complex Systems*. Constantine, Algérie, pp. 149–159. (Visited on 02/08/2012).

Dong, Xin Luna, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, pp. 601–610. ISBN: 9781450329569. DOI: 10.1145/2623330.2623623.

Ehrig, Marc and Steffen Staab (2004). "QOM - Quick Ontology Mapping". In: *3rd International Semantic Web Conference (ISWC04)*. Springer, pp. 683–697.

Ehrlenspiel, Klaus and Harald Meerkamm (2013). *Integrierte Produktentwicklung - Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Carl Hanser Verlag München Wien.

Eigner, Martin and Ralph Stelzer (2009a). "Der Produktentstehungsprozess im Wandel". In: *Product Lifecycle Management*. Springer-Verlag Berlin Heidelberg. Chap. 2, pp. 9–25.

– (2009b). "PLM als Backbone der virtuellen Produktentstehung". In: *Product Lifecycle Management*. Springer-Verlag Berlin Heidelberg. Chap. 4, pp. 47–63.

Eigner, Martin and Ralph Stelzer (2009c). *Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management*. Berlin: Springer Berlin Heidelberg.

Eisenbarth, Thomas (2013). "Semantic Process Models - Transformation, Adaption, Resource Consideration". PhD thesis. Augsburg, Germany: University of Augsburg.

Ensan, Faezeh (2010). "Semantic Interface-Based Modular Ontology Framework". PhD thesis. The University of New Brunswick.

Ernzer, M. and Herbert Birkhofer (2002). "Selecting Methods for Life Cycle Design Based on the Needs of a Company". In: *DESIGN 2002 - 7th International Design Conference*, pp. 1305–1310.

Farooq, Amjad and M Junaid Arshad (2010). "A Process Model for Developing Semantic Web Systems". In: *New York Science Journal* 3 (9), pp. 34–39.

Feigenbaum, Edward A and Pamela McCorduck (1983). *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Addison-Wesley, p. 275. ISBN: 0201115190.

Fellbaum, Christiane, ed. (1998). *WordNet: An Electronic Lexical Database*. Language, speech, and communication. Cambridge, MA, USA: MIT Press.

Fink, Michael (2011). *D2.6 - Consistency Maintenance. Final Report*. ONTORULE Project.

Fischer, Wolf (2013). "Linguistically Motivated Ontology-Based Information Retrieval". PhD thesis. Augsburg, Germany: University of Augsburg.

Foo, G and D.J. Friedman (1992). "Variability and Capability: The Foundation of Competitive Operations Performance". In: *AT&T Technical Journal* (Jul/Aug), pp. 2–9.

Forgy, Charles L. (1982). "Rete: A fast algorithm for the many pattern/many object pattern match problem". In: *Artificial Intelligence* 19 (1), pp. 17–37. ISSN: 00043702. DOI: 10.1016/0004-3702(82)90020-0.

Fowler, Martin (1996). *Analysis Patterns - Reusable object models*. Addison-Wesley Professional.

Frank, Ulrich (2002). "Multi-perspective enterprise modeling (MEMO) conceptual framework and modeling languages". In: *Hawaii International Conference on System Sciences*.

– (2014). "Multi-perspective enterprise modeling: Foundational concepts, prospects and future research challenges". In: *Software and Systems Modeling* 13 (3), pp. 941–962. ISSN: 16191374. DOI: 10.1007/s10270-012-0273-9.

Franke, H.J., S. Löffler, and M. Deimel (2003). "The database 'Methodos' assists an effective application of design methods". In: *14th International Conference on Engineering Design (ICED 2003)*. Stockholm, pp. 155–156.

Fürber, Christian and Martin Hepp (2010). "Using SPARQL and SPIN for Data Quality Management on the Semantic Web". In: *Business Information Systems*. Ed. by Witold Abramowicz and Robert Tolksdorf. Vol. 47. Lecture Notes in Business Information Processing (LNBIP). Springer-Verlag Berlin Heidelberg, pp. 35–46.

Georgakopoulos, Diimitrios, Mark Hornick, and Amit Sheth (1995). "An overview of workflow management: From process modeling to workflow automation infrastructure". In: *Distributed and Parallel Databases* 3 (2), pp. 119–152. (Visited on 02/07/2014).

Gerber, Aurona, Paula Kotzé, and Alta Van der Merwe (2010). "Towards the Formalisation of the TOGAF Content Metamodel using Ontologies". In: *12th International Conference on Enterprise Information Systems (ICEIS)*. Ed. by Joaquim Filipe and José Cordeiro. Madeira, Portugal: SciTePress, pp. 54–64.

Ghawi, Raji and Nadine Cullot (2007). "Database-to-Ontology Mapping Generation for Semantic Interoperability". In: *Third International Workshop on Database Interoperability*. ISBN: 9781595936493.

Giapoulis, A. (1998). *Modelle für effektive Konstruktionsprozesse*. Aachen: Shaker.

Gleichauf, Bettina (2011). "Planung der Unternehmensarchitektur". PhD thesis. Universität St. Gallen.

Grau, Bernardo Cuenca, Bijan Parsia, and Evren Sirin (2009). "Ontology Integration Using $\mathcal{E}$-Connections". In: *Modular Ontologies*. Ed. by Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra. Springer Berlin/Heidelberg. Chap. Connecting, pp. 293–320.

Greiffenberg, Steffen (2003). "Methodenbewertung mittels Quality Function Deployment". In: *MobIS*. Ed. by Elmar J. Sinz, Markus Plaha, and Peter Neckel. Bamberg: GI, pp. 131–153.

Gruber, Thomas (1993). "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5 (April), pp. 199–220.

– (2008). "Ontology". In: *Encyclopedia of Database Systems*. Ed. by Ling Liu and M. Tamer Özsu. Springer-Verlag.

Grüninger, Michael and M Fox (1995). "Methodology for the Design and Evaluation of Ontologies". In: *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*.

Guarino, Nicola (1997). "Semantic matching: formal ontological distinctions for information organization, extraction, and integration". In: *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Lecture Notes in Computer Science, vol. 1299*, pp. 139–170. ISBN: ISBN 3-540-63438-X. DOI: 10.1007/3-540-63438-X_8.

– (1998). "Formal Ontology and Information Systems". In: *Fois'98* 46 (June), pp. 3–15.

Guarino, Nicola and Christopher A Welty (2009). "An Overview of OntoClean". In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. International Handbooks on Information Systems. Springer. Chap. 10, pp. 201–220. ISBN: 3540408347. DOI: 10.1007/978-3-540-92673-3.

Halpin, Harry, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson (2010). "When owl : sameAs isn't the Same : An Analysis of Identity in Linked Data". In: *The Semantic Web–ISWC 2010*, pp. 305–320. ISBN: 9783642177453. DOI: 10.1007/978-3-642-17746-0.

Hammer, M. and J. Champy (1993). *Reengineering the corporation: A manifesto for business revolution*. New York, New York, USA: HarperBusiness.

Hanschke, Inge (2013). *Strategisches Management der IT-Landschaft*. Munich, Germany: Carl Hanser Verlag München, p. 634.

Hayes, Robert H. and Steven C. Wheelwright (1984). *Restoring Our Competitive Edge: Competing Through Manufacturing*. Wiley. ISBN: 0471051594.

Herfeld, Ulrich (2007). "Matrix-basierte Verknüpfung von Komponenten und Funktionen zur Integration von Konstruktion und numerischer Simulation". PhD thesis. Munich, Germany: TU München.

Hess, Claudia, Willy Chen, and Thomas Syldatke (2008). "Business-oriented CAx Integration with Semantic Technologies". In: *3rd International Applications of Semantic Technologies Workshop located at the Informatik 2008*. Munich.

– (2010). "Business-oriented CAx Integration with Semantic Technologies Revisited". In: *5th International Applications of Semantic Technologies Workshop*. Leipzig.

Hess, Claudia, Julian Lambertz, and Thomas Syldatke (2009). "Ontologien im Praxiseinsatz: Erfahrungen bei der Audi AG". In: *Objektspektrum* 06, pp. 47–52.

Hess, Claudia, Florian Lautenbacher, and Katrin Fehlner (2013). "Business building blocks as coordination mechanism for enterprise transformations". In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, pp. 194–203. ISSN: 15417719. DOI: 10.1109/EDOCW.2013.29.

Hess, Thomas (1996). *Entwurf betrieblicher Prozesse, Grundlagen - Bestehende Methoden - Neue Ansätze*. Wiesbaden: Deutscher Universitäts-Verlag.

Hesse, Wolfgang (2002). "Ontologie(n)". In: *Informatik Spektrum* (25), pp. 477–480.

Hesse, Wolfgang, G. Merbeth, and R. Frölich (1992). "Software-Entwicklung - Vorgehensmodelle, Projektführung und Produktverwaltung". In: *Handbuch der Informatik* 5.3.

Hitzler, Pascal, Markus Krötzsch, and Sebastian Rudolph (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall / CRC. ISBN: 978-1420090505.

Honke, Benjamin (2013). "Situational Method Engineering for the Enactment of Method-Centric Domain-Specific Languages". PhD thesis. Augsburg, Germany: University of Augsburg.

Hoppenbrouwers, Jeroen, Sjir Nijssen, and Koen Van Leeuwen (2012). *D1.6 - Final Business Modeling Methodology VOLUME I*. ONTORULE Project.

Horridge, Matthew, Nick Drummond, John Goodwin, Alan Rector, and Hai H Wang (2006). "The Manchester OWL Syntax". In: *OWLED2006 Second Workshop on OWL Experiences and Directions*. Athens, GA, USA.

Horrocks, Ian, Peter F Patel-Schneider, Frank Van Harmelen, and Frank van Harmelen (2003). "From SHIQ and RDF to OWL: the making of a Web Ontology Language". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1 (1), pp. 7–26. ISSN: 15708268. DOI: 10.1016/j.websem.2003.07.001.

Horrocks, Ian, Ulrike Sattler, and Stephan Tobies (2000). "Practical Reasoning for Expressive Description Logics". In: *Logic Journal Of The Igpl*. Lecture Notes in Artificial Intelligence 8 (1705). Ed. by H Ganzinger, D McAllester, and A Voronkov, pp. 161–180.

ISO (1987). *ISO/TR 9007: Information processing systems - Concepts and terminology for the conceptual schema and the information base*.

– (2014). *ISO 10303-242:2014 - Industrial automation systems and integration – Product data representation and exchange – Part 242: Application protocol: Managed model-based 3D engineering*.

Janik, Maciej, Ansgar Scherp, and Steffen Staab (2011). "The Semantic Web: Collective Intelligence on the Web". In: *Informatik-Spektrum* 34 (5), pp. 469–483. ISSN: 0170-6012. DOI: 10.1007/s00287-011-0535-x. (Visited on 11/03/2011).

Kattenstroth, Heiko, Wolfgang May, and Franz Schenk (2007). "Combining OWL with F-logic rules and defaults". In: *CEUR Workshop Proceedings* 287, pp. 60–75. ISSN: 16130073.

Kendall, Elisa, Roy Bell, Roger Burkhart, Mark Dutra, and Evan Wallace (2009). "Towards a Graphical Notation for OWL 2". In: *OWLED 09*. Vol. 2009.

Kifer, Michael, Georg Lausen, and James Wu (1995). "Logical Foundations of Object-Oriented and Frame-Based Languages". In: *Journal of ACM* 42 (1995), pp. 741–843. DOI: 10.1145/210332.210335.

Kiss, Eva Maria, Patrick Albert, Roman Korf, Peter Rosina, Jeroen Hoppenbrouwers, and Sjir Nijssen (2010). *D8.4 - Market Intelligence Report*. ONTORULE Project.

Korf, Roman, Eva Maria Kiss, Patrick Albert, et al. (2009). *D8.3 - Exploitation Plan*. ONTORULE Project.

Korf, Roman, Eva Maria Kiss, Florine Durand, Martin Doušek, Adil El Ghali, Amina Chniti, Diego Berrueta, and Francois Lévy (2010). *D2.4 - Extended demonstration prototypes for dissemination , teaching and public experimentation purposes*.

Krötzsch, Markus, Frederick Maier, Adila a. Krisnadhi, and Pascal Hitzler (2011). "A Better Uncle for OWL - Nominal schemas for integrating rules and description logics". In: *20th International Conference on the World Wide Web (WWW-11)*. Hyderabad, India, pp. 645–654. ISBN: 9781450306324. DOI: 10.1145/1963405.1963496.

Lambrix, Patrick and He Tan (2007). "A Tool for Evaluating Ontology Alignment Strategies". In: *Journal on Data Semantics* 182, pp. 182, 202.

Langermeier, Melanie, Thomas Driessen, Peter Rosina, and Bernhard Bauer (2014). "Change and Version Management in Variability Models for Modular Ontologies". In: *16th International Conference on Enterprise Information Systems (ICEIS)*. Lisbon, Portugal.

Langermeier, Melanie, Peter Rosina, Heiner Oberkampf, Thomas Driessen, and Bernhard Bauer (2013). "Management of Variability in Modular Ontology Development". In: *Service-Oriented Computing - ICSOC 2013 Workshops*. Ed. by Alessio Lomuscio, Surya Nepal, Fabio Patrizi, Boualem Benatallah, and Ivona Brandic. Lecture Notes in Computer Science. Berlin, Germany: Springer, pp. 225–239.

Li, Man, Dazhi Wang, Xiaoyong Du, and Shan Wang (2005). "Ontology Construction for Semantic Web : A Role-based Collaborative Development Method". In: *Proceedings of the 7th Asia-Pacific web conference on Web Technologies Research and Development, APWeb05*. Shanghai, China: Springer-Verlag, pp. 609–619. DOI: 10.1007/978-3-540-31849-1_60.

Liebig, Thorsten, Marko Luther, Olaf Noppens, and Michael Wessel (2011). "OWLlink". In: *Semantic Web – Interoperability, Usability, Applicability* 2 (1), pp. 23–32.

Lindemann, Udo (2002). "Flexible Adaption of Methods within the Design Process". In: *7th International Design Conference – DESIGN 2002*. Dubrovnik, Croatia, pp. 81–86.

– (2009). "Vorgehensmodelle, Grundprinzipien und Methoden". In: *Methodische Entwicklung technischer Produkte*. Garching: Springer-Verlag Berlin Heidelberg. Chap. 3.

Lindström, John, Daria Plankina, Hakan Lideskog, Magnus Löfstrand, and Lennart Karlsson (2013). "Functional Product Development: Criteria for Selection of Design Methods on Strategic and Operational Levels". In: *The Philosopher's Stone for Sustainability*. Ed. by Yoshiki Shimomura and Koji Kimita. Tokyo, Japan: Springer Berlin Heidelberg, pp. 25–30.

Löffler, Stefan and Burkhard Jagusch (2004). "'Methodos' - ein Methodenbaukasten zur effizienten Produktentwicklung". In: *Industrieerprobte Lösungen und Werkzeuge für Produktentwicklung, Engineering und Kompetenzmanagement*. Braunschweig, pp. 175–188.

López-Mesa, Belinda (2003). "Selection and use of engineering design methods using creative problem solving". PhD thesis. Luleå, Sweden: Luleå University of Technology.

Marcos, Gorka et al. (2005). "A Semantic Web based approach to multimedia retrieval". In: *Proceedings of the fourth International Workshop on Content-based Multimedia Indexing (CBMI 2005)*. Riga, Latvia. ISBN: 9521513640. (Visited on 12/05/2013).

Marino, O., F. Rechenmann, and P. Uvietta (1990). "Multiple Perspectives and Classification Mechanism in Object-Oriented Representation". In: *Proc. of ECAI'90*. Stockholm, pp. 425–430.

Matsokis, Aristeidis (2010). "An Ontology-Based Approach for Closed-Loop Product Lifecycle Management". PhD thesis. Lausanne, Switzerland: École Polytechnique Fédérale de Lausanne.

Matthes, D (2011). *Enterprise Architecture Frameworks Kompendium*. Springer Berlin Heidelberg. ISBN: 978-3-642-12954-4. DOI: 10.1007/978-3-642-12955-1.

McGuinness, Deborah L, Richard Fikes, James Rice, and Steve Wilder (2000). "An Environment for Merging and Testing Large Ontologies". In: *Proceedings of the*

Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), pp. 483–493.

McGuinness, Deborah L and Jon R. Wright (1998). "Conceptual modelling for configuration: A description logic-based approach". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing - Special Issue on Configuration Design* 12 (4), pp. 333–344.

Meerkamm, Harald (2011). "Methodology and Computer-Aided Tools - a Powerful Interaction for Product Development". In: *The future of design methodology*. Ed. by Herbert Birkhofer. Springer London Limited. Chap. 5, pp. 55–65.

Miller, J and J Mukerji (2003). "MDA Guide Version 1.0. 1". In: *Object Management Group* (June). (Visited on 02/07/2014).

Möller, Knud (2012). "Lifecycle models of data-centric systems and domains". In: *Semantic Web journal*.

Motta, E. (1999). *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*. Amsterdam, The Netherlands: IOS Press. ISBN: 1586030035.

Mueller, Johannes (1990). *Arbeitsmethoden der Technikwissenschaften: Systematik, Heuristik, Kreativität*. Springer Verlag.

Musen, Mark A (1992). "Dimensions of knowledge sharing and reuse". In: *Computers and Biomedical Research* 25, pp. 435–467.

Nazarenko, Adeline, Abdoulaye Guissé, Francois Lévy, Nouha Omrane, Sylvie Szulman, Jeroen Hoppenbrouwers, Sjir Nijssen, and Koen Van Leeuwen (2012). *D1.6 - Final Business Modeling Methodology VOLUME II*. ONTORULE Project.

Nguyen, Vinh, Olivier Bodenreider, and Amit Sheth (2014). "Don't like RDF reification? Making Statements about Statements using Singleton Property". In: *23rd international conference on World wide web (WWW14)*. Geneva, Switzerland, pp. 759–770. ISBN: 9781450327442. DOI: 10.1145/2566486.2567973. (Visited on 06/18/2014).

Nijssen, Sjir, Koen Van Leeuwen, Jeroen Hoppenbrouwers, and Hennie Bouwmeester (2012). *D1.7 - Conceptual (Enterprise) Modeling*. ONTORULE Project.

Nonaka, Ikujiro (2007). "The Knowledge-Creating Company". In: *Harvard Business Review* (July).

Oberhauser, Roy and Rainer Schmidt (2007). "Towards a Holistic Integration of Software Lifecycle Processes using the Semnatic Web". In: *2nd Int. Conf. on Software and Data Technologies (ICSOFT'07)*. Barcelona, Spain, pp. 137–144.

Object Management Group (2008). *Software and Systems Process Engineering Meta-Model Specification*.
– (2009). *Ontology Definition Metamodel*.
– (2011). *OMG Meta Object Facility (MOF) Core Specification*.
– (2014). *Decision Model and Notation*.
Omrane, Nouha, Adeline Nazarenko, Peter Rosina, Sylvie Szulman, and Christoph Westphal (2011). "Lexicalized Ontology for a Business Rules Management Platform: An Automotive Use Case". In: *Rule - Based Modeling and Computing on the Semantic Web*. Ed. by Frank Olken, Monica Palmirani, and Davide Soltara. Vol. 7018. Springer Berlin Heidelberg, pp. 179–192. DOI: 10.1007/978-3-642-24908-2_21.
Omrane, Nouha, Adeline Nazarenko, and Sylvie Szulman (2011). "Les entités nommées : éléments pour la conceptualisation". In: *22èmes Journées Francophones d'Ingénierie des Connaissances (IC"1)*.
Ortmann, Jens, Philipp Diefenthaler, Florian Lautenbacher, Claudia Hess, and Willy Chen (2014). "Unternehmensarchitekturen mit Semantischen Technologien". In: *HMD Praxis der Wirtschaftsinformatik* 51 (5), pp. 616–626.
Österle, Hubert (1995). *Business Engineering. Prozess- und Systementwicklung: Band 1: Entwurfstechniken*. Berlin: Springer.
Ouziri, Mourad (2009). "Perspective-based accessing to Web datasources". In: *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009* 3, pp. 706–710. DOI: 10.1109/ICICISYS.2009.5358429.
Ovtcharova, J. (2009). *'Virtual Engineering'*. Lecture. Karlsruhe, Germany.
Pahl, Gerhard, Wolfgang Beitz, Jörg Feldhusen, and K.-H. Grote (2007). *Engineering Design – A Systematic Approach*. Ed. by K. Wallace and L. Blessing. London: Springer-Verlag London Limited.
Pahl, Gerhard, Wolfgang Beitz, Jörg Feldhusen, and K.H. Grote (2007). *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*. Springer Berlin/Heidelberg.
Parreiras, Fernando Silva, Tobias Walter, Christian Wende, and Edward Thomas (2010). "Model Driven Development with Semantic Web Technologies". In: *6th European Conference on Modeling Foundations and Applications ECMFA 2010*. Paris, France.
Patel-Schneider, Peter F, P Hayes, Ian Horrocks, and Frank Van Harmelen (2004). *OWL Web Ontology Language Semantics and Abstract Syntax - W3C Recommendation*.

Pedrinaci, Carlos and John Domingue (2009). "Ontology-based metrics computation for business process analysis". In: *4th International Workshop on Semantic Business Process Management*. ACM New York, NY, USA, pp. 43–50. DOI: 10.1145/1944968.1944976. (Visited on 06/04/2014).

Potter, Stephen (2003). *A Survey of Knowledge Acquisition from Natural Language*. AKT Project. Edinburgh, Scotland.

Poulikidou, Sofia (2012). *Literature review Methods and tools for environmentally friendly product design and development Identification of their relevance to the vehicle design context*. Stockholm, Sweden: Department of Urban Planning, Environment School of Architecture, and the Built Environment KTH, Royal Institute of Technology.

Probst, Gilbert, Steffen Raub, and Kai Romhardt (2012). *Wissen managen*. Vol. 35. Wiesbaden: Springer Gabler.

Reich, Yoram (2010). "My method is better!" In: *Research in Engineering Design* 21 (3), pp. 137–142. ISSN: 09349839. DOI: 10.1007/s00163-010-0092-3.

Ribière, Myriam and Rose Dieng-Kuntz (2002). "A viewpoint model for cooperative building of an ontology". In: *Conceptual Structures: Integration and Interfaces*, pp. 220–234. (Visited on 02/08/2012).

Rodríguez, Jesús Barrasa, Jesús Barrasa Rodriguez, and Asunción Gómez-Pérez (2006). "Upgrading relational legacy data to the semantic web". In: *Proceedings of the 15th international conference on World Wide Web WWW 06* 20 (September 1998), p. 1069. ISSN: 08963207. DOI: 10.1145/1135777.1136019.

Rosina, Peter and Bernhard Bauer (2015). "Enterprise Methods Management using Semantic Web Technologies". In: *Fifth International Symposium on Business Modeling and Software Design (BMSD)*. Ed. by Boris Shishkov. Milan, Italy: Scitepress. ISBN: 979-989-758-111-3.

Rosina, Peter and Eva Maria Kiss (2011). *D4.3 - AUDI R&D Business Orchestration System*. ONTORULE Project.

Rosina, Peter and Thomas Syldatke (2010). *D4.2 - Semantic integration of BOMs - public demonstrator*. ONTORULE Project.

– (2011). *D4.4 - Evaluation of the AUDI R&D Business Orchestration System*. ONTORULE Project.

Roth, D., H. Binz, and R. Watty (2010). "Generic structure of knowledge within the product development". In: *DESIGN 2010*. Glasgow: Design Society, pp. 1681–1690.

Salvadores, Manuel, Matthew Horridge, Paul R. Alexander, Ray W. Fergerson, Mark A Musen, and Natalya F. Noy (2012). "Using SPARQL to query bioportal ontologies and metadata". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7650 LNCS (PART 2), pp. 180–195.

Schreiber, Guus, Hans Akkermans, Anjo A. Anjewierden, Robert Dehoog, Nigel R. Shadbolt, Walter Van de Velde, and Bob J. Wielinga (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, MA, USA: MIT Press. ISBN: 978-0-262-19300-9.

Shaw, Marianne, Landon T Detwiler, Natalya Fridman Noy, James F Brinkley, and Dan Suciu (2011). "vSPARQL: a view definition language for the semantic web." In: *Journal of Biomedical Informatics* 44 (1), pp. 102–117.

Shvaiko, Pavel and Jerome Euzenat (2005). "A Survey of Schema-based Matching Approaches". In: *Journal on Data Semantics*.

– (2013). "Ontology Matching: State of the Art and Future Challenges". In: *IEEE Transactions on Knowledge and Data Engineering* 25 (1), pp. 158–176. ISSN: 10414347. DOI: 10.1109/TKDE.2011.253.

Sirin, Evren, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz (2007). "Pellet: A practical OWL-DL reasoner". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (2), pp. 51–53. ISSN: 15708268. DOI: 10.1016/j.websem.2007.03.004.

Sowa, John F. (1976). "Conceptual Graphs for a Data Base Interface". In: *IBM Journal of Research and Development* 20 (4), pp. 336–357. ISSN: 0018-8646. DOI: 10.1147/rd.204.0336.

– (2010). "Integrating Semantic Systems". In: *SemTech 2010*. San Francisco, CA, USA.

Staab, Steffen and Rudi Studer, eds. (2009). *Handbook on Ontologies*. Springer. ISBN: 978-3540709992.

Staab, Steffen, Tobias Walter, Gerd Gröner, and Fernando Silva Parreiras (2010). "Model Driven Engineering with Ontology Technologies". In: *Lecture Notes in Computer Science* 6325 (Reasoning Web. Semantic Technologies for Software Engineering. L), pp. 62–98.

Stadler, Rupert (2011). "Audi - the dawning of a new age of automobility". In: *Keynote speech at the AmCham Business Luncheon*. Frankfurt: Audi Communications.

Stuckenschmidt, Heiner (2006). "Toward multi-viewpoint reasoning with owl ontologies". In: *The Semantic Web: Research and Applications*, pp. 259–272. (Visited on 02/08/2012).

Studer, R., R. Benjamins, and D. Fensel (1998). "Knowledge Engineering: Principles and Methods". In: *Data & Knowledge Engineering* 25 (1-2), pp. 161–197.

Sunkle, Sagar, Vinay Kulkarni, and Suman Roychoudhury (2013). "Analyzing Enterprise Models Using Enterprise Architecture-Based Ontology". In: *MODELS 2013*. Ed. by Ana Moreira, Bernhard Schätz, Jeff Gray, Antonio Vallecillo, and Peter Clarke. LNCS. Miami, FL, USA: Springer Berlin Heidelberg, pp. 622–638.

Sure, York, Steffen Staab, and Rudi Studer (2009). "Ontology Engineering Methodology". In: *Handbook on Ontologies*. Ed. by Steffen Staab and Rudi Studer. International Handbooks on Information Systems. Springer Berlin Heidelberg, pp. 135–152. ISBN: 9783540709992. DOI: 10.1007/978-3-540-92673-3.

Syldatke, Thomas, Willy Chen, Jürgen Angele, and Andreas Nierlich (2007). "How Ontologies and Rules Help to Advance Automobile Development". In: *International Symposium, RuleML 2007*. Orlando, Florida, USA: Springer Berlin Heidelberg.

Syldatke, Thomas, Marcus Lutz, Willy Chen, and Claudia Hess (2008). "Managing Dependencies in the Product Development Process with Semantic Technologies". In: *10th International Design Structure Matrix Conference, DSM'08*. Stockholm, Sweden, pp. 7–9.

The Open Group (2011). *TOGAF® Version 9.1*. Zaltbommel, Netherlands: Van Haren Publishing, p. 654. ISBN: 9789087536794.

– (2013). *ArchiMate® 2.1 Specification*. Berkshire, United Kingdom: The Open Group. ISBN: 1-937218-43-0.

Traczyk, Wiesław (2005). "Structural representations of unstructured knowledge". In: *Journal of Telecommunications and Information Technology* (3), pp. 81–86. (Visited on 11/26/2014).

Tsarkov, Dmitry and Ian Horrocks (2006). "FaCT ++ Description Logic Reasoner : System Description". In: *Automated Reasoning. Lecture Notes in Computer Science* 4130, pp. 292–297. DOI: 10.1007/11814771_26.

United Nations Economic Commision for Europe (2009). *ECE Regulation No. 16 Agreement, Revision 6*.

Uschold, Mike, Martin King, Stuart Moralee, and Yannis Zorgios (1998). "The Enterprise Ontology". In: *The Knowledge Engineering Review* 13 (01), pp. 31–89.

Verma, Kunal and Alex Kass (2010). "Model-Assisted Software Development : Using a ' semantic bus ' to automate steps in the software development process". In: *Semantic Web* 1 (1-2), pp. 17–24. DOI: 10.3233/SW-2010-0022.

Wallace, K. (2011). "Transferring Design Methods into Practice". In: *The future of design methodology*. Ed. by Herbert Birkhofer. Springer London Limited. Chap. 21, pp. 239–248.

Weber, Christian (2005). "CPM / PDD - An extended theoretical approach to modelling products and product development processes". In: *GermanIsraeli Symposium on Advances in Methods and Systems for Development of Products and Processes* 7. Ed. by H Bley, H Jansen, F L Krause, and M Shpitalni, pp. 159–179.

– (2011). "Design Theory and Methodology – Contributions to the Computer Support of Product Development/Design Processes". In: *The future of design methodology*. Ed. by Herbert Birkhofer. Chap. 8, pp. 91–104.

Wegmann, Alain (2003). "On the Systemic Enterprise Architecture Methodology (SEAM)". In: *5th International Conference on Enterprise Information Systems (ICEIS)*, pp. 483–490.

Wegmann, Alain, Gil Regev, Irina Rychkova, Lam-Son Lê, Jose Diego de la Cruz, and Philippe Julia (2007). "Business-IT Alignment with SEAM for Enterprise Architecture". In: *The 11th IEEE International EDOC Conference (EDOC)*. Annapolis, Maryland, USA. ISBN: 0-7334-2276-4.

Weigt, Markus (2008). "Systemtechnische Methodenentwicklung". PhD thesis. Karlsruhe, Germany: Universität Karlsruhe (TH). ISBN: 9783866442856.

Weinberger, Danny (2010). "... und am Anfang steht die Geschäftsanforderung , oder ?" In: *OBJEKTspektrum* (EAM).

Westphal, Christoph, Harald Meerkamm, Sandro Wartzack, and Kristin Paetzold (2009). "New ways of data processing for increasing of the efficiency within the product development". In: *International Conference on Engineering Design, ICED'09*. Stanford, CA, pp. 105–114.

Westphal, Christoph and Sandro Wartzack (2010). "Ontologiebasierte Generierung, Visualisierung und Analyse von Informationsketten in der Produktentwicklung". In: *21. Symposium Design for X*. Ed. by Dieter Krause, Kristin Paetzold, and Sandro Wartzack. Hamburg: TuTech Verlag, pp. 143–155.

– (2011). "Integrated process and product model for the evaluation of product properties". In: *18th International Conference on Engineering Design, ICED'11*. Copenhagen, Denmark, pp. 538–549.

Wood, David, ed. (2010). *Linking Enterprise Data*. Springer. ISBN: 9781441976642.

Zachman, J. a. (1987). *A framework for information systems architecture*. DOI: 10.1147/sj.263.0276.

Zedlitz, Jesper, Jan Jörke, Norbert Luttenberger, and J Jan (2012). "From UML to OWL 2". In: *Communications in Computer and Information Science*. Ed. by D. Lukose, A.R. Ahmad, and A. Suliman. Springer, Heidelberg, pp. 154–163. DOI: 10.1007/978-3-642-32826-8_16.

Zillner, Sonja, Heiner Oberkampf, Peter Rosina, Melanie Langermeier, and Thomas Driessen (2015). "Method of Composing an Integrated Ontology". US20150081648A1.

Zuo, Landong (2006). "A semantic and agent-based approach to support information retrieval, interoperability and multi-lateral viewpoints for heterogeneous environmental databases". PhD thesis. Queen Mary, University of London.

Online Sources

Autodesk (2007). *Autodesk® -Lösungen für Mechanik und Maschinenbau. Digital Prototyping: Fragen und Antworten*. URL: http://marvo.li/CAD/digital_prototyping_faqs.pdf (visited on 08/10/2015).

Bechhofer, Sean, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and Lynn Andrea Stein (2004). *OWL Web Ontology Language Reference*. URL: http://www.w3.org/TR/owl-ref/.

Beckett, David (2014). *RDF 1.1 N-Triples*. W3C. URL: http://www.w3.org/TR/n-triples/ (visited on 02/16/2015).

Berners-Lee, Tim (2006). *Artificial Intelligence and the Semantic Web*. URL: http://www.w3.org/2006/Talks/0718-aaai-tbl/ (visited on 08/10/2015).

Berners-Lee, Tim and Dan Connolly (2011). *Notation3 (N3): A readable RDF syntax*. URL: http://www.w3.org/TeamSubmission/n3/ (visited on 11/22/2012).

Blumauer, Andreas (2014). *Why SKOS should be a Focal Point of your Linked Data Strategy*. Semantic Web Company. URL: http://de.slideshare.net/semwebcompany/why-skos-should-be-a-focal-point-of-your-linked-data-strategy (visited on 07/22/2015).

BMIR (2015). *Protégé*. Stanford Center for Biomedical Informatics Research (BMIR). URL: http://protege.stanford.edu/ (visited on 07/14/2015).

Boley, Harold and Michael Kifer (2013). *RIF Basic Logic Dialect (Second Edition)*. W3C. URL: http://www.w3.org/TR/rif-bld/ (visited on 07/28/2015).

Bourque, P. and R.E. Fairley, eds. (2014). *Guide to the Software Engineering Body of Knowledge*. URL: http://www.swebok.org/ (visited on 08/06/2015).

Brickley, Dan and Libby Miller (2014). *FOAF Vocabulary Specification*. URL: http://xmlns.com/foaf/spec/ (visited on 04/20/2015).

Business Rules Group (2000). *Defining Business Rules ~ What Are They Really?* URL: http://www.businessrulesgroup.org/first_paper/br01c0.htm (visited on 06/23/2015).

– (2003). *The Business Rules Manifesto*. URL: http://www.businessrulesgroup.org/brmanifesto.htm (visited on 07/28/2015).

Businessrules.ch (2008). *Business Rules Lifecycle*. URL: http://www.businessrules.ch/wp-content/uploads/2008/04/br-schema.jpg (visited on 05/09/2011).

Casanave, Cory (2012). *Semantic Information Modeling for Federation*. URL: http://www.omg.org/news/meetings/tc/dc-12/special-events/pdf/Casanave-SIMF.pdf (visited on 09/01/2014).

D'Arcus, Bruce and Frédérick Giasson (2009). *Bibliographic Ontology Specification*. URL: http://bibliontology.com/ (visited on 08/04/2014).

Davis, Ian (2005). *VANN: A vocabulary for annotating vocabulary descriptions*. URL: http://purl.org/vocab/vann/ (visited on 08/04/2014).

de Sainte Marie, Christian, Gary Hallmark, and Adrian Paschke (2013). *RIF Production Rule Dialect (Second Edition)*. W3C. URL: http://www.w3.org/TR/rif-prd/ (visited on 07/28/2015).

Dublin Core Metadata Initiative (2014). *dublincore.org*. URL: http://dublincore.org/ (visited on 05/13/2014).

Duden (2015). *'Methode'*. URL: http://www.duden.de/rechtschreibung/Methode (visited on 07/06/2015).

EABOK Consortium (2014). *Enterprise Architecture Body of Knowledge*. URL: http://www2.mitre.org/public/eabok/index.html (visited on 07/21/2014).

Epistemics (2003). *Knowledge Acquisition*. URL: http://www.epistemics.co.uk/Notes/63-0-0.htm (visited on 07/14/2015).

Food and Agriculture Organization of the United Nations (2014). *Geopolitical Ontology*. URL: http://www.fao.org/countryprofiles/geoinfo/en/ (visited on 08/05/2014).

FORUM Working Group (2008). *General Syntactic Changes Relative to the Original F-logic Syntax*. URL: http://forum.projects.semwebcentral.org/forum-syntax.html (visited on 01/01/2015).

Gandon, Fabien and Guus Schreiber, eds. (2014). *RDF 1.1 XML Syntax*. W3C. URL: http://www.w3.org/TR/rdf-syntax-grammar/ (visited on 02/16/2015).

Giasson, Frédérick and Michael Bergman (2015). *Upper Mapping and Binding Exchange Layer (UMBEL) Specification*. URL: http://techwiki.umbel.org/index.php/UMBEL_Specification (visited on 04/01/2015).

Gougeon, Alain (2003). *Everything you ever wanted to know about Business Rules*. Proyecto ILACO II. URL: http://ww.bptrends.com/publicationfiles/07-03%20ART%20Everything%20About%20Bus%20Rules%20-%20Gougeon.pdf (visited on 10/15/2013).

Harris, Steve and Andy Seaborne (2013). *SPARQL 1.1 Query Language*. URL: http://www.w3.org/TR/sparql11-query/ (visited on 05/22/2013).

Herman, Ivan, Ben Adida, Manu Sporny, and Mark Birbeck, eds. (2015). *RDFa 1.1 Primer - Third Edition*. W3C. URL: http://www.w3.org/TR/rdfa-primer/ (visited on 10/27/2015).

Horrocks, Ian, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C. URL: http://www.w3.org/Submission/SWRL/ (visited on 07/28/2015).

Iparraguirre, Pedro (2013). *Produktentwicklung mit virtuellen Prototypen*. University of Siegen. URL: https://wiki.zimt.uni-siegen.de/fertigungsautomatisierung/index.php/Produktentwicklung_mit_virtuellen_Prototypen (visited on 08/10/2015).

Isaac, Antoine and Ed Summers (2009). *SKOS Simple Knowledge Organization System Primer*. URL: http://www.w3.org/TR/skos-primer/ (visited on 04/20/2015).

Jacobs, Stephan (2014). *Reifegradmodelle*. Lehrstuhl für Wirtschaftsinformatik, Europa-Universität Viadrina Frankfurt (Oder). URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/reifegradmodelle (visited on 04/15/2014).

Kifer, Michael and Harold Boley (2013). *RIF Overview (Second Edition)*. W3C. URL: http://www.w3.org/TR/rif-overview/.

Knublauch, Holger (2014). *SPARQL Web Pages (SWP, aka UISPIN)*. Topquadrant. URL: http://uispin.org/ (visited on 06/01/2015).

Knublauch, Holger, James Hendler, and Kingsley Idehen (2011). *SPIN - Overview and Motivation*. URL: http://www.w3.org/Submission/spin-overview/ (visited on 10/10/2013).

Lehmann, Jens et al. (2014). *DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia*. URL: http://semantic-web-journal.net/system/files/swj499.pdf (visited on 06/16/2014).

McGuinness, Deborah L and Frank van Harmelen (2004). *OWL Web Ontology Language Overview*. URL: http://www.w3.org/TR/owl-features/.

Meißner, Jörg D. (2006). *Allgemeine Hinweise zur Bewertung von Methoden (in verschiedenen Anwendungsbereichen)*. Institut für Verkehrswirtschaft, Straßenwesen und Städtebau, Leibniz Universität Hannover. URL: http://www.optiv.de/OptimVerkehr/Allgemeine_Hinweise/allgemeine_hinweise.pdf (visited on 07/06/2015).

Merriam-Webster.com (2015). *'method'*. URL: http://www.merriam-webster.com/dictionary/method (visited on 04/20/2015).

Motik, Boris (2015). *KAON2*. URL: http://kaon2.semanticweb.org/ (visited on 02/19/2015).

Motik, Boris, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz (2009). *OWL 2 Web Ontology Language Profiles*. Ed. by B Motik, B Cuenca Grau, I Horrocks, Z Wu, A Fokoue, and C Lutz. The World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/owl2-profiles/ (visited on 11/10/2015).

Motik, Boris, Peter F Patel-Schneider, Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Ulrike Sattler (2009). *OWL 2 Web Ontology Language Direct Semantics*. Ed. by Boris Motik, Peter F Patel-Schneider, and Bernardo Cuenca Grau. The World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/.

Nielsen, Michael A. (2015). *Neural Networks and Deep Learning*. URL: http://neuralnetworksanddeeplearning.com/ (visited on 08/18/2015).

Noy, Natalya Fridman and Deborah L McGuinness (2000). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University. URL: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

Object Management Group (2015). *Semantics of Business Vocabulary and Business Rules (SBVR)*. Object Management Group. URL: http://www.omg.org/spec/SBVR/ (visited on 07/28/2015).

ontoprise GmbH (2007). *How to write F-Logic-Programs*. URL: http://www.semafora-systems.com/documents/tutorial_flogic.pdf (visited on 01/01/2015).

ONTORULE (2011). *OWL*. ONTORULE Project. URL: http://ontorule-project.eu/showcase/OWL (visited on 02/18/2015).

OptiV (2006). *Methodenauswahl*. OptiV - Erschließung von Entscheidungs- und Optimierungsmethoden für die Anwendung im Verkehr. URL: http://www.optiv.de/Methoden/ModMetho/pages/node11.htm (visited on 06/15/2015).

Oxford Dictionaries (2015). *method*. URL: http://www.oxforddictionaries.com/definition/english/method (visited on 04/20/2015).

Parreiras, Fernando Silva, Tobias Walter, Christian Wende, and Edward Thomas (2012). *Model-Driven Software Development with Semantic Web Technologies*. URL:

http://de.slideshare.net/fparreiras/modeldriven-software-development-with-semantic-web-technologies (visited on 05/11/2012).

Polikoff, Irene (2011). *Ontologies and Data Models - are they the same?* URL: http://topquadrantblog.blogspot.de/2011/09/ontologies-and-data-models-are-they.html (visited on 05/11/2013).

Progrenium (2015). *Das Dilemma mit der Vielfalt*. URL: http://www.progenium.com/Publikationen/DE/data/upload/publikation/PROGENIUM_Pressemitteilung_Das%20Dilemma%20mit%20der%201424941380.pdf (visited on 03/25/2015).

Prud'hommeaux, Eric and Gavin Carothers, eds. (2014). *RDF 1.1 Turtle*. W3C. URL: http://www.w3.org/TR/2014/REC-turtle-20140225/ (visited on 02/16/2015).

Schreiber, Guus and Yves Raimond, eds. (2014). *RDF 1.1 Primer*. W3C. URL: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/ (visited on 02/16/2015).

semafora systems (2012). *ObjectLogic Tutorial*. URL: http://www.semafora-systems.com/fileadmin/user_upload/Publications_EN/ObjectLogic_Tutorial.pdf (visited on 02/19/2015).

Sporny, Manu, Dave Longley, Gregg Kellog, Markus Lanthaler, and Niklas Lindström (2014). *JSON-LD 1.0*. W3C. URL: http://www.w3.org/TR/json-ld/ (visited on 02/16/2015).

The Apache Software Foundation (2014). *Reasoners and rule engines: Jena inference support*. URL: https://jena.apache.org/documentation/inference/ (visited on 06/23/2014).

Thomsen, Thomas (2013). *ASAM ODS*. URL: https://wiki.asam.net/display/STANDARDS/ASAM+ODS (visited on 05/12/2013).

Vatant, Bernard and Marc Wick (2012). *GeoNames Ontology*. URL: http://www.geonames.org/ontology/ (visited on 08/05/2014).

W3C (2013). *W3C Semantic Web Activity*. URL: http://www.w3.org/2001/sw/ (visited on 02/22/2013).

– (2007). *Semantic Web layercake diagram*. URL: http://www.w3.org/2007/03/layerCake.png (visited on 03/25/2015).

W3C OWL Working Group (2012). *OWL 2 Web Ontology Language Document Overview*. Ed. by Boris Motik, Peter F Patel-Schneider, and Bijan Parsia. The World Wide Web Consortium (W3C). URL: http://www.w3.org/TR/owl2-overview/ (visited on 05/09/2013).

W3PM (2009). *Product Modelling using Semantic Web Technologies*. W3C. URL: http://www.w3.org/2005/Incubator/w3pm/XGR-w3pm/ (visited on 06/21/2014).

WordNet (2013a). *'method'*. URL: http://wordnetweb.princeton.edu/ (visited on 04/23/2013).

– (2013b). *'technique'*. URL: http://wordnetweb.princeton.edu/ (visited on 04/23/2013).

Glossary

ABox assertional box.
ACMM Architecture Capability Maturity Model.
ADM TOGAF Architecture Development Method.
AHP Analytic Hierarchy Process.
ALUO Advantages-Limitations-Uniqueness-Opportunities.
API Application Programming Interface.
AR Augmented Reality.
ASAM Association for Standardisation of Automation and Measuring Systems.
ATF ASAM transport format.
AVx Individual software for the processing of orders and test preparations.

BAL Business Action Language.
BBB Business Building Block.
bibo Bibliographic Ontology Specification.
BoK Body of Knowledge.
BOM Bill of Material.
BPA Business Process Analysis.
BPEL Business Process Execution Language.
BPM Business Process Management.
BPMN Business Process Model and Notation.
BR business rule.
BRL Business Rules Language.
BRMS Business Rule Management System.

CA Computer Aided.
CAD Computer Aided Design.
CAE Computer Aided Engineering.
CAME Computer Aided Method Engineering.
CAT Computer Aided Testing.
CAx Computer Aided x.
CFD Computational Fluid Dynamics.
CiDaD Competence in Design and Development.
CMM Capability Maturity Model.
CMMI Capability Maturity Model Integration.

CNL Controlled Natural Language.
CPM Characteristics-Properties Modeling.
CRUD Create, Read, Update, Delete.
CSV Comma Separated Value.
CV Controlled Vocabulary.
CWA Closed World Assumption.

DAML DARPA Agent Markup Language.
DARPA US Defense Advanced Research Projects Agency.
DC Dublin Core.
DDL Distributed Description Logics.
DE Domain Expert.
DL Description Logic.
DMN Decision Model And Notation.
DMU Digital MockUp.
DO domain ontology.
DoC Department of Commerce.
DP Digital Prototype.
DSL Domain Specific Language.
DTM Design Theory and Methodology.

EA Enterprise Architecture.
EABOK® Enterprise Architecture Body of Knowledge.
EAF Enterprise Architecture Framework.
EAM Enterprise Architecture Management.
ECU Electronic Control Unit.
ERP Enterprise Resource Planning.
ESC Electronic Stability Control.

F-Logic Frame Logic.
FEM Finite Element Method.
FOAF Friend of a Friend.
FOL First-Order Logic.

GPL General Purpose Language.
GWT Google Web Toolkit.

HIC Head Injury Criterion.
HiL Hardware in the Loop.

HoQ House of Quality.
HT Highlighting Technique.

IBF Interface-Based modular ontology Formalism.
iPeM Integrated Product-Engineering Model.
IRI Internationalized Resource Identifier.
ISO International Organization for Standardization.
IT Information Technology.

KA Knowledge Acquisition.
KB Knowledge Base.
KE Knowledge Engineer.
KM Knowledge Management.
KO Knowledge Owner.
KPI Key Performance Indicator.

LP Logic Programming.

MBS Multi Body Simulation.
MEMO Multi-perspective Enterprise MOdeling.
MGR Manager.
MMM Munich Model of Methods.
MOF Meta Object Facility.
MVP Model-view-presenter.

NAF Negation as failure.
NLP Natural Language Processing.
NVH 'Noise, Vibration, Harshness'.

ODS Open Data Services.
OEM Original Equipment Manufacturer.
OIL Ontology Inference Layer.
OMG Object Management Group.
OWA Open World Assumption.
OWL Web Ontology Language.

PCP Product Creation Process.
PDD Property-Driven Development.
PDL Package Based Description Logics.
PDM Product Data Management.

PDP Product Development Process.
PLM Product Lifecycle Management.
PoMM Process-oriented Method Model.

QFD Quality Function Deployment.

R&D Research & Development.
R&WM Rating & Weighting Method.
RDF Resource Description Framework.
RDFa Resource Description Framework in Attributes.
RDFS Resource Description Framework Schema.
RIF Rule Interchange Format.
ROI Return on investment.

SBVR Semantics of Business Vocabulary and Business Rules.
SEAM Systemic Enterprise Architecture Methodology.
SiL Software in the Loop.
SIMF Semantic Information Modeling for Federation.
SKOS Simple Knowledge Organization System.
SME Small and medium enterprise.
SOP Start of Production.
SPARQL SPARQL Protocol And RDF Query Language.
SPEM Software and Systems Process Engineering Meta-Model.
SPICE Software Process Improvement and Capability Determination.
SPIN SPARQL Inferencing Notation.
STEP STandard for the Exchange of Product model data.
SW Semantic Web.
SWEBOK Software Engineering Body of Knowledge.
SWP SPARQL Web Pages.
SWRL Semantic Web Rule Language.
SWT Semantic Web Technology.

TBox terminological box.
TOGAF The Open Group Architecture Framework.

UI User Interface.
UML Unified Modeling Language.
UofD Universe of Discourse.
URI Uniform Resource Identifier.

VR Virtual Reality.

W3C World Wide Web Consortium.
W3PM W3C Product Modelling Incubator Group.
WSM Weighted sum model.
WWW World Wide Web.

XML Extensible Markup Language.
XSD XML Schema Definition.

List of Figures

1.1. Development of the product strategy in the automotive industry. . 5
1.2. Numerous dependencies between body components and loading cases illustrating the complexity in product development. 6
1.3. Lifecycle juxtaposition of business, domain knowledge, processes, rules and applications. 9
1.4. A theoretical model of the structure of knowledge in the product development. 23
1.5. Thesis structure. 28

2.1. Various KA techniques. 33
2.2. Semantic Web Architecture. 35
2.3. Relations between kinds of ontologies. 38
2.4. Assertional box (ABox) and Terminological box (TBox) compared to the Meta Object Facility (MOF). 41
2.5. An Resource Description Framework (RDF) graph depicting two nodes connected by a triple. 44
2.6. The structure of OWL 2. 46
2.7. The ONTORULE platform: illustrating three main activities and separation between ontologies and rules. 55
2.8. The different phases of a product, i.e., the product lifecycle and design work. 56
2.9. Shifting of result critical sub-processes and resources towards early phases of development. 58
2.10. Enterprise Architecture domains. 61
2.11. The Open Group Architecture Framework (TOGAF) Content Metamodel. 63
2.12. TOGAF Architecture Development Method (ADM) Cycle. 65
2.13. Correspondence between ArchiMate (including extensions) and TOGAF. 66

3.1. Process-oriented Method Model (PoMM). 71
3.2. Method building blocks. 72
3.3. The *Munich Model of Methods (MMM)*. 72

3.4. Exemplary screenshot of the Competence in Design and Development (CiDaD) portal. . . . 74
3.5. *Methodos*: Method synectics and stored information (Franke, S. Löffler, and Deimel 2003). . . . 75
3.6. Method Selection and Combination in a Product Creation Process (PCP). . . . 77
3.7. Software and Systems Process Engineering Meta-Model (SPEM)'s conceptual usage framework. . . . 79
3.8. CAx architecture layers. . . . 81
3.9. Visualization of a CAx architecture ontology. . . . 82
3.10. Excerpt of a method meta model for method engineering. . . . 83
3.11. Preselecting conceptual design methods after Franke, S. Löffler, and Deimel (2003). . . . 86

4.1. An overview of various design methods. . . . 106
4.2. An overview of various tools. . . . 106
4.3. A conceptual model of the method concept and its relations. . . . 108
4.4. A classification of the four design-relevant activity areas together with goals. . . . 111
4.5. Processes and design methods in a PDP. . . . 112
4.6. Structure of the core method ontology. . . . 114
4.7. Method formulations for "Specifying a task" as a flowchart and template. . . . 117
4.8. Concrete vs. Abstract Methods. . . . 123
4.9. Method Metrics Ontology. . . . 124
4.10. Method ontology extract with maturities. . . . 125
4.11. Information architecture example. . . . 127
4.12. "A retained classification of elementary schema-based matching approaches" (Shvaiko and Euzenat 2005). . . . 129
4.13. Reification example, expressing the certainty about an RDF statement. . . . 130
4.14. An exemplary alignment between two simple ontologies. . . . 131
4.15. Complex Match between Digital MockUp (DMU) and AVx Ontology. . . . 132
4.16. Classification of database-to-ontology mapping approaches. . . . 135
4.17. Adapting SQL databases with SWT. . . . 136
4.18. The method ontology's architecture. . . . 138

4.19. Concrete and Abstract Resources. 140
4.20. Product Models. 141
4.21. A selection of document types for the method ontology. 144
4.22. SKOS example. 146

5.1. Relevant part of the ONTORULE platform for our business case. . 154
5.2. Methodology. 159

6.1. EA ontology based on TOGAF and ArchiMate meta models. 163
6.2. Meta models at different specificity levels. 164
6.3. Mapping of EA and core method ontology. 166
6.4. The process modeling extension. 169
6.5. An example of multiple viewpoints for a car's description. 171
6.6. Overview of the personae developed during the ONTORULE project. 173
6.7. Example of the links between multiple experts and multiple viewpoints in an application. 175
6.8. Starting points for method selection. 181
6.9. Interactions between the meta model, building blocks, diagrams and stakeholders. 184
6.10. Capabilities Increments and Dimensions. 188
6.11. Capability Increment "Radar". 189

7.1. A lexicalized ontology for annotating source documents. 205
7.2. CAx Methods and Tools. 213
7.3. Integration of BOMs. 216
7.4. Reduction of development time by using DMU. 217
7.5. Mapping virtual and physical BOMs with mapping ontology and rules. 218
7.6. Structure of the BOM part number. 220
7.7. Extracts of DMU and AVx BOM ontologies. 221
7.8. BOM demonstrator 3-Tier architecture. 225
7.9. BOM Matcher showing statistical analysis result in pie charts. . . . 227
7.10. BOM Matcher showing different structure trees and mapping rules. 228
7.11. Evaluated thesis structure elements for the BOM case study. 232
7.12. The Association for Standardisation of Automation and Measuring Systems (ASAM) Open Data Services (ODS) base elements . . 235

7.13. Possible mapping between ASAM ODS and Method ontology (extract). . . . 236
7.14. Simplified extract of the case study ontology. . . . 241
7.15. Process model of the product property validation. . . . 242
7.16. Example of an importable Excel sheet. . . . 249
7.17. PDD Schema. . . . 250
7.18. Architecture. . . . 252
7.19. Ontology Import Schema. . . . 253
7.20. PDD demonstrator landing page. . . . 254
7.21. Evaluated thesis structure elements for the PDD case study. . . . 267

8.1. A theoretical model of the structure of knowledge in the product development. . . . 270

B.1. PDD demonstrator method chains. . . . 321
B.2. Property Spider. . . . 322
B.3. Information Chain. . . . 322
B.4. Product Information. . . . 322
B.5. Quality View. . . . 323
B.6. System Analysis – Process Analysis. . . . 323
B.7. Free Ontology Browsing. . . . 324
B.8. Basis Schema. . . . 328
B.9. Assessment Schema. . . . 330
B.10. Analysis Schema. . . . 332
B.11. ModelCreation Schema. . . . 334

List of Tables

2.1. Synonyms. 43

3.1. Comparison of meta model features. 91
3.2. Comparison of meta model elements. 93

4.1. Exemplary questions regarding a method application and environment. 100
4.2. Overview and classification of exemplary methods and tools. 102
4.3. Method assessment aspects and criteria. 120

6.1. Method Architecture Artifacts Catalogs. 185
6.2. Method Architecture Artifacts Catalogs for Data Quality. 186
6.3. Method Architecture Artifacts Matrices of Class Relations. 187
6.4. Method Architecture Artifacts Diagrams. 188

7.1. Mapping Categories. 222

A.1. DMU concepts. 317
A.2. AVx concepts. 318

B.1. *AA_Basis* concepts descriptions. 329
B.2. *01_Assessment* concepts descriptions. 331
B.3. *02_Analysis* concepts descriptions. 332
B.4. *03_ModelCreation* concepts descriptions. 333

Listings

2.1. "An informal example of Resource Description Framework Schema (RDFS) triples" (Schreiber and Raimond 2014). 44
2.2. F-Logic Schema. 47
2.3. F-Logic Facts. 48
2.4. F-Logic Rules. 48
2.5. F-Logic Queries. 48
2.6. An example rule in OWL Description Logic (DL). 50
2.7. Example rule that is not expressible in DL. 50
2.8. Example rule of an e-commerce shop system. 51

4.1. Rename property with SPARQL *CONSTRUCT*. 133
4.2. Rename class with SPARQL *CONSTRUCT*. 133
4.3. Transform and filter values with SPARQL *CONSTRUCT*. 134
4.4. Expressing scales and units in a product ontology. 143
4.5. Multiple *rdfs:label* annotations. 146
4.6. Using SKOS to organize knowledge. 147
4.7. Concept as *skos:Concept* and *owl:Class*. 147
4.8. Linking SKOS and OWL via OWL annotation property. 148

6.1. Example of a SPARQL query for receiving preferred label properties. 191

7.1. Rule example, that calculates the total process duration. 203
7.2. Formalized example concept `BuckleTest` and including SKOS extension. 205
7.3. ObjectLogic query for receiving `Tools` and `Product Information`, related by a `Process` and `Method`. 206
7.4. SKOS preferred and alternative concept labels. 207
7.5. Concept definition by applying `<skos:definition>`. 208
7.6. Example SPARQL query for finding concepts that include a specific text. 210
7.7. Querying for subconcepts. 210
7.8. A fragment of text annotated by the lexicalized ontology. 211

A.1. "Mirrored part number in AVx"-Rule. 319

A.2. "Mirrored part in AVx"-Rule. 319
A.3. "AVx part number"-Rule. 319

B.1. "AnalysisResult is based on Analysis"-Rule. 325
B.2. "Uses AnalysisResult"-Rule. 325
B.3. "Estimated or Actual Cost"-Rule. 325
B.4. "Planner Matrix"-Rule. 325
B.5. "Planner Matrix"-Query. 325
B.6. "Planner Matrix transPI"-Rule 1. 326
B.7. "Planner Matrix transPI"-Rule 2. 326
B.8. "ModelCreation is executed at Milestone"-Rule. 326
B.9. "Helper information chain"-Rule. 326
B.10. "Not managed product information"-Query. 327

A

BOM Mapping Case Study

A.1. Ontologies

The following concepts represent the ontologies needed for the mapping of the DMU BOM combined with the AVx BOM. Most of the properties and some concepts are not needed for the actual mapping that is why they are left out in the listing below. Nevertheless, the non-mentioned properties are part of the ontologies that have been used in the use cases. They consist of information required by the engineers and technicians who prepare and test virtual and physical vehicles.

Concept	Property	Type
`StructurePart`	hasDMUPart	#Part
	hasParentStructure	#StructurePart
`Part`	partNumber	String
	PDA	String
	Alternative	String
	Version	String
	Date	String
	Naming	String

Table A.1.: DMU concepts.

Concept	Property	Type
`StructureNode`	Naming	String
	hasParentStructure	#StructureNode
`Usage`	hasStructureNode	#StructureNode
	hasUsage	#Usage
	partInfoA	String
	partInfoB	String
`Part`	partInfoC	String
	partInfoD	String
	partInfoE	String
	drawingDate	String
	quantity	String

Table A.2.: AVx concepts.

A.2. Rules and Queries

```
RULE #partNumberMirroredPartAVx: FORALL ↙
    ↳ aPart1,aPart2,DI1,DI2,V,M,DI1no,DI2no,result
  aPart1[#partNumberMirroredPart->aPart2]
<-
  aPart1:avx#Part AND
  aPart1[avx#partInfoC->DI1] AND
  aPart2:avx#Part AND
  aPart2[avx#partInfoC->DI2] AND
  aPart1[avx#partInfoD->V] AND
  aPart2[avx#partInfoD->V] AND
  aPart1[avx#partInfoE->M] AND
  aPart2[avx#partInfoE->M] AND
  string2number(DI1,DI1no) AND
  NOT even(DI1no) AND
  string2number(DI2,DI2no) AND
  even(DI2no) AND
  add(DI1no,1.0,result) AND
  isequal(result,DI2no).
```

Listing A.1: "Mirrored part number in AVx"-Rule.

```
RULE #mirroredPartAVx: FORALL aPart1,aPart2
  aPart1[#mirroredPartAVx->aPart2]
<-
  aPart1:avx#Part AND
  aPart1[#partNumberMirroredPart->aPart2] AND
  aPart2:avx#Part AND
  NOT aPart1[#isStandardPart->true] AND
  NOT aPart1[#isDeveloperPart->true] AND
  aPart1[#siblingAVx->aPart2].
```

Listing A.2: "Mirrored part in AVx"-Rule.

```
RULE #setPartNumberAVx: FORALL aPart,PN,a,b,c,d,e,e2,r1,r2,r3
  aPart[#partNumber->PN]
<-
  aPart:avx#Part AND
  aPart[avx#partInfoA->a] AND
```

```
6       aPart[avx#partInfoB->b] AND
        aPart[avx#partInfoC->c] AND
8       aPart[avx#partInfoD->d] AND
        aPart[avx#partInfoE->e] AND
10      replace(e," ","",e2) AND
        concat(a,b,r1) AND
12      concat(r1,c,r2) AND
        concat(r2,d,r3) AND
14      concat(r3,e2,PN).
```

Listing A.3: "AVx part number"-Rule.

B

PDD Case Study

In this chapter, the ontologies as well as the User Interfaces (UIs) used for the PDD case study (*cf.* section 7.5) are listed.

B.1. User Interface

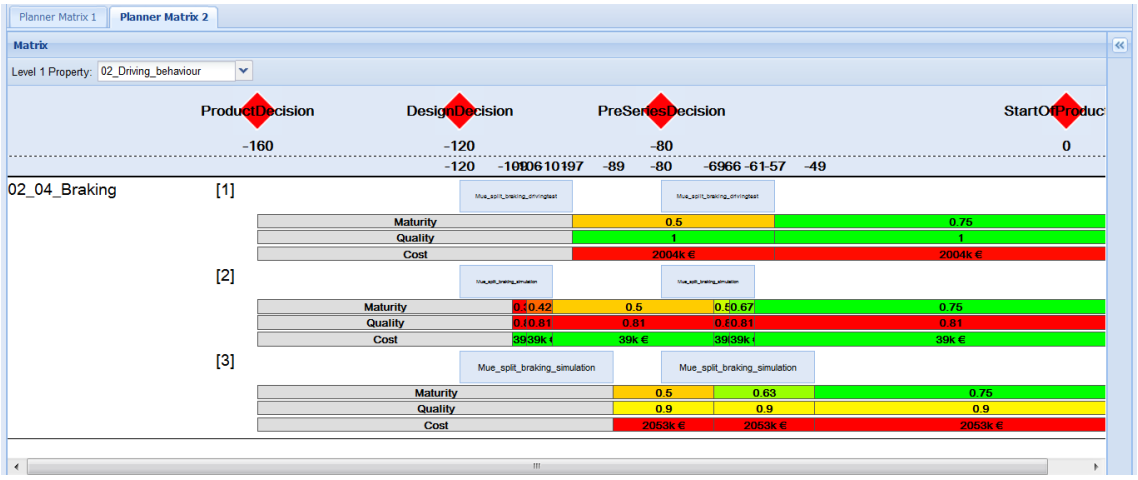


Figure B.1.: PDD demonstrator method chains.

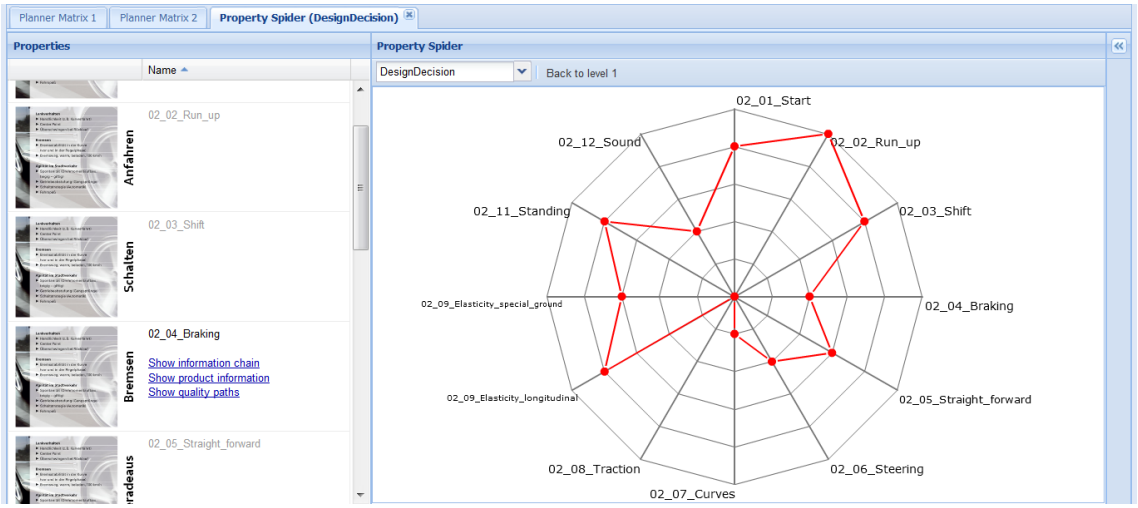


Figure B.2.: Property Spider.

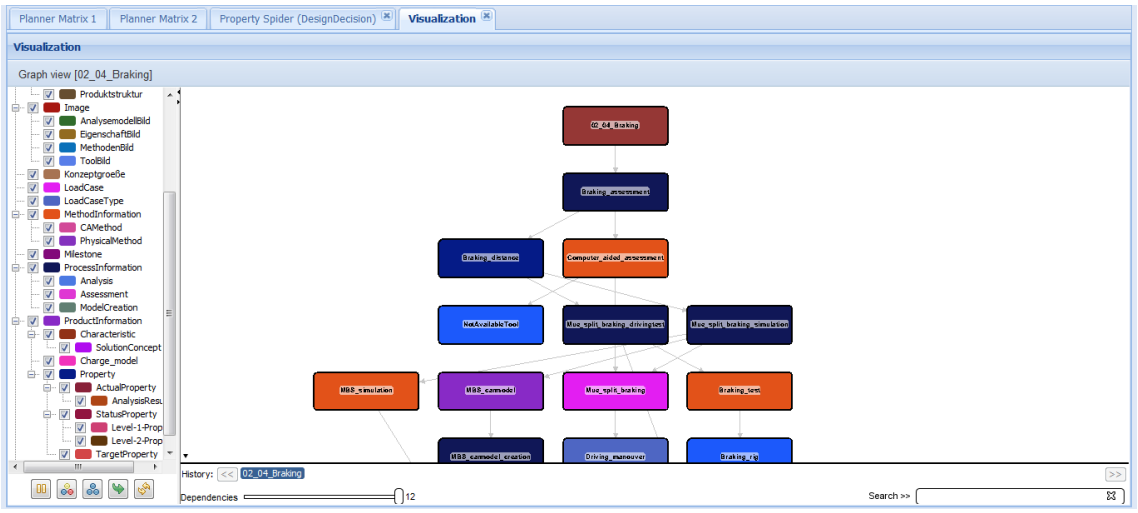


Figure B.3.: Information Chain.

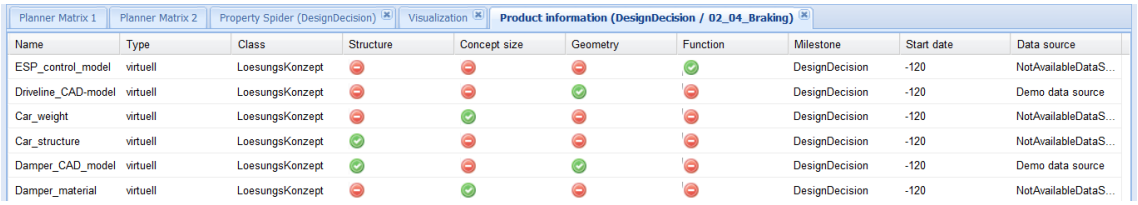


Figure B.4.: Product Information.

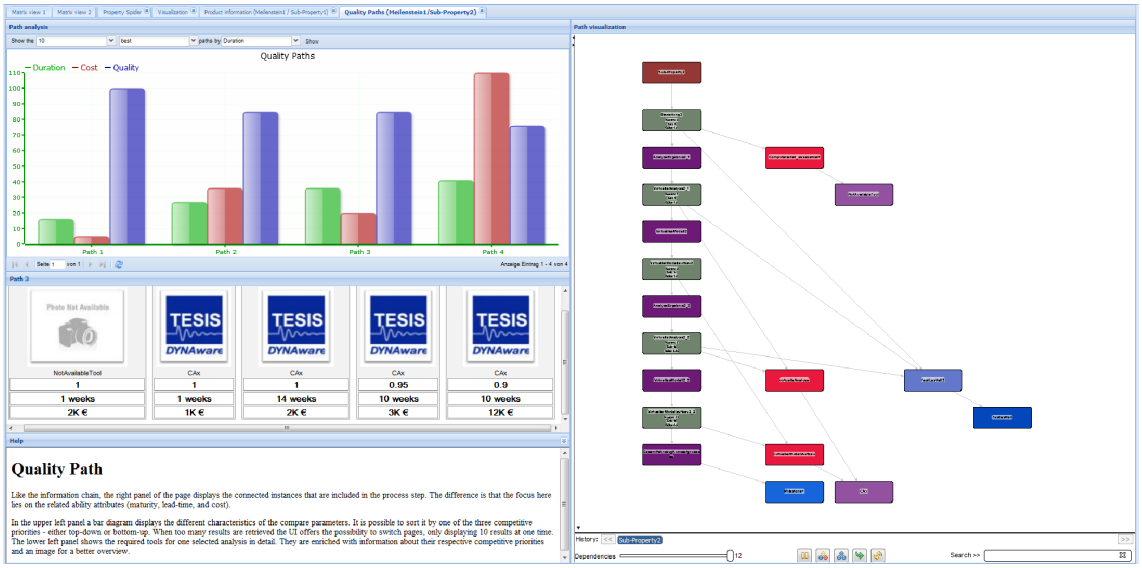


Figure B.5.: Quality View.

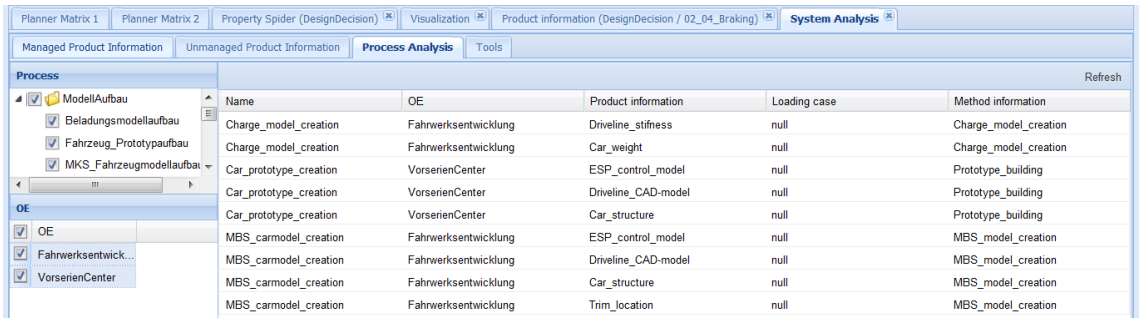


Figure B.6.: System Analysis – Process Analysis.

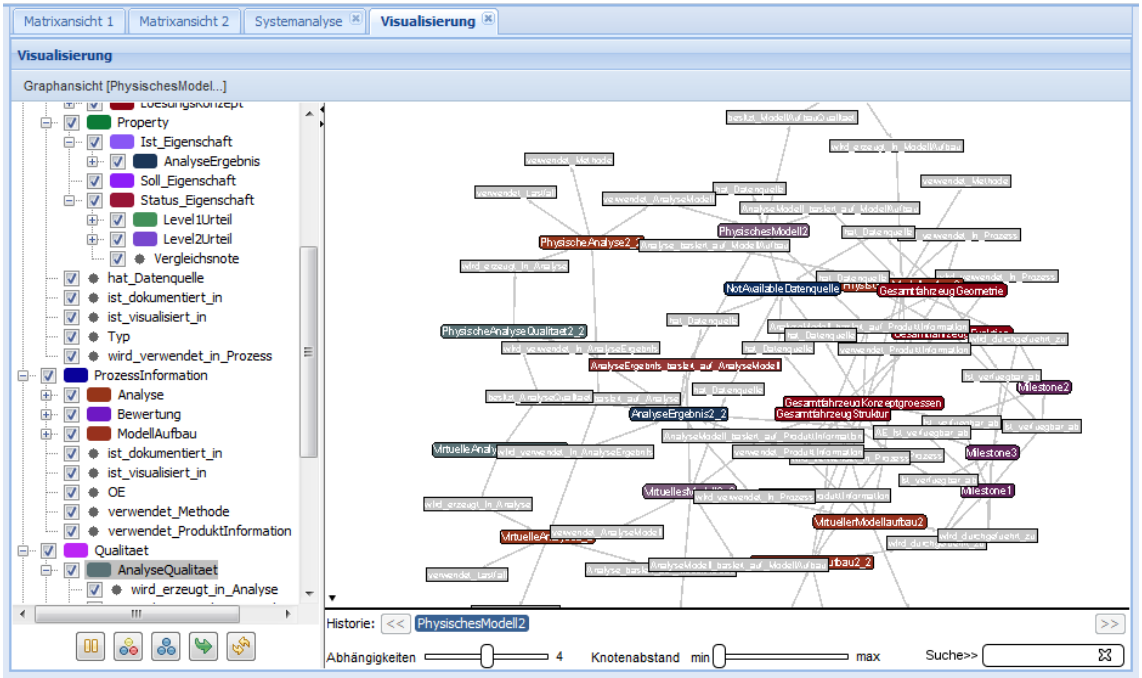


Figure B.7.: Free Ontology Browsing.

B.2. Rules and Queries

```
?A[AnalysisResult_is_based_on_Analysis->?F]
:-
?A:AnalysisResult@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?A[owns_analysis_ability ->?D] AND ↙
    ↳ ?D:AnalysisAbility@<http://www.example.com/02_Analysis> AND
?D[is_created_in_analysis->?F] AND ↙
    ↳ ?F:Analysis@<http://www.example.com/02_Analysis>.
```

Listing B.1: "AnalysisResult is based on Analysis"-Rule.

```
?A[AnalysisResult_is_based_on_Analysis->?F]
:-
?A:AnalysisResult@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?A[owns_analysis_ability ->?D] AND ↙
    ↳ ?D:AnalysisAbility@<http://www.example.com/02_Analysis> AND
?D[is_created_in_analysis->?F] AND ↙
    ↳ ?F:Analysis@<http://www.example.com/02_Analysis>.
```

Listing B.2: "Uses AnalysisResult"-Rule.

```
?G[cost->?H] :- ?G:Ability AND ?G[Estimated_Cost->?H] AND ?G[Actual_Cost->-1.0].
?G[cost->?H] :- ?G:Ability AND ?G[Actual_Cost->?H] AND NOT _unify(?H,-1.0).
```

Listing B.3: "Estimated or Actual Cost"-Rule.

```
f_helper_Planner_Matrix(?Level1Property, ?Level2Property, ?Milestone, ?Totalcost, ↙
    ↳ ?Totaltime, ?Totalquality, ?Totalmaturity, ?AnalysisProcesses)
:-
?#AH:Helper_Planner_Level1Properties_AnalysisResult[ Level1Property-> ↙
    ↳ ?Level1Property, Level2Property->?Level2Property, AACost->?AACost, ↙
    ↳ AATime->?AATime, AAQuality->?AAQuality, AnalysisResult->?AnalysisResult] ↙
    ↳ AND ?SolutionConcept:SolutionConcept AND ↙
    ↳ f_helper_Planner_Matrix_transPI(?AnalysisResult, ?SolutionConcept, ↙
    ↳ ?PI_PI_Cost, ?PI_PI_Time, ?PI_PI_Quality, ?AnalysisProcesses, ?Milestone) ↙
    ↳ AND (?Totalcost = ?PI_PI_Cost+?AACost) and (?Totaltime = ↙
    ↳ ?PI_PI_Time+?AATime) and (?Totalquality = ?PI_PI_Quality*?AAQuality) and ↙
    ↳ ?Milestone[Maturity->?Maturity] AND (?Totalmaturity = ?Maturity * ↙
    ↳ ?Totalquality).
```

Listing B.4: "Planner Matrix"-Rule.

```
@{options[sort(?Level1Property, ?Level2Property, ?Milestone, ?AnalysisProcesses, ↙
    ↳ ?Totaltime, ?Totalcost),outorder(?Level1Property, ?Level2Property, ↙
    ↳ ?Milestone, ?Totalcost, ?Totaltime, ?Totalquality, ?Totalmaturity, ↙
    ↳ ?AnalysisProcesses), EvaluationMethod(BottomUp)]}
?-
f_helper_Planner_Matrix(?Level1Property, ?Level2Property, ?Milestone, ?Totalcost, ↙
    ↳ ?Totaltime, ?Totalquality, ?Totalmaturity, ?AnalysisProcesses).
```

Listing B.5: "Planner Matrix"-Query.

```
f_helper_Planner_Matrix_transPI(?AnalysisResult, ?SolutionConcept, ?Cost, ?Time, ↙
    ↳ ?Quality, [?Analysis], ?Milestone)
:-
?ModelCreation:ModelCreation[executed_at->?Milestone] AND ?Milestone:Milestone ↙
    ↳ AND ?#H:Helper_PI_PI_Abilities[PI->?AnalysisResult, PI2->?SolutionConcept, ↙
    ↳ Cost->?Cost, Time->?Time, Quality->?Quality, Analysis->?Analysis, ↙
    ↳ ModelCreation->?ModelCreation] AND ? AnalysisResult:AnalysisResult AND ↙
    ↳ ?SolutionConcept:SolutionConcept AND ?Analysis:Analysis.
```

Listing B.6: "Planner Matrix transPI"-Rule 1.

```
f_helper_Planner_Matrix_transPI(?AnalysisResult1, ?AnalysisResult2, ?Totalcost, ↙
    ↳ ?Totaltime, ?Totalquality, ?Analyses, ?MSList)
:-
?ModelCreation:ModelCreation AND ?#H:Helper_PI_PI_Abilities[PI->?AnalysisResult1, ↙
    ↳ PI2->?AnalysisResult2, Cost->?Cost, Time->?Time, Quality->?Quality, ↙
    ↳ Analysis->?Analysis1, ModelCreation->?ModelCreation] AND ↙
    ↳ ?AnalysisResult1:AnalysisResult AND ?AnalysisResult2:AnalysisResult AND ↙
    ↳ ?Analysis1:Analysis AND f_helper_Planner_Matrix_transPI(?AnalysisResult2, ↙
    ↳ ?ProductInformation, ?preCost, ?preTime, ?preQuality, ?Analysis2, ?MSList) ↙
    ↳ AND ?Totalcost = ?Cost + ?preCost and ?Totaltime = ?Time + ?preTime and ↙
    ↳ ?Totalquality = ?Quality * ?preQuality AND ?Analyses=[?Analysis1, ↙
    ↳ ?Analysis2].
```

Listing B.7: "Planner Matrix transPI"-Rule 2.

```
?ModelCreation[executed_at->?Milestone]
:-
?ModelCreation:ModelCreation@<http://www.example.com/03_Model Creation> AND ↙
    ↳ ?ModelCreation[uses_ProductInformation->?SC] AND ↙
    ↳ ?SC:SolutionConcept@<http://www.example.com/03_ModelCreation> AND ↙
    ↳ ?SC[is_available_at->?Milestone] AND ↙
    ↳ ?Milestone:Milestone@<http://www.example.com/ 03_ModelCreation> AND ↙
    ↳ ?Milestone[StartDate->?SD] AND ?MaxMinSD = max{?SD2 [?ModelCreation] | ↙
    ↳ ?ModelCreation:ModelCreation@<http://www.example.com/03_ ModelCreation> ↙
    ↳ AND ?ModelCreation[uses_ProductInformation->?SC2] AND ↙
    ↳ ?SC2:SolutionConcept@<http://www.example.com/03_ ModelCreation> AND ↙
    ↳ ?SC2[min_available_at(?MS2)->?SD2] AND ↙
    ↳ ?MS2:Milestone@<http://www.example.com/03_ModelCreation>} AND ?SD>=?MaxMinSD.
```

Listing B.8: "ModelCreation is executed at Milestone"-Rule.

```
@{InformationChain_Rule}
f_helper_InformationChain(?AnalysisResult, ?Analysis, ?LoadCase, ?LoadCaseType, ↙
    ↳ ?MethodInformation1, ?Tool1, ?AnalysisModel, ?ModelCreation, ↙
    ↳ ?MethodInformation2, ?Tool2, ?ProductInformation) : ↙
    ↳ Helper_InformationChain[InformationChain_AnalysisResult -> ↙
    ↳ ?AnalysisResult, InformationChain_Analysis->?Analysis, ↙
    ↳ InformationChain_LoadCase->?LoadCase, ↙
    ↳ InformationChain_LoadCaseType->?LoadCaseType, ↙
    ↳ InformationChain_MethodInformation1->?MethodInformation1, ↙
    ↳ InformationChain_Tool1->?Tool1, ↙
    ↳ InformationChain_AnalysisModel->?AnalysisModel, ↙
    ↳ InformationChain_ModelCreation->?ModelCreation, ↙
```

```
    ↳ InformationChain_MethodInformation2->?MethodInformation2, ↙
    ↳ InformationChain_Tool2->?Tool2, ↙
    ↳ InformationChain_ProductInformation->?ProductInformation]
:-
?AnalysisResult:AnalysisResult@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?AnalysisResult[AnalysisResult_based_on_Analysis -> ?Analysis] AND ↙
    ↳ ?Analysis:Analysis@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?MethodInformation1[uses_Tool -> ?Tool1] AND ↙
    ↳ ?MethodInformation1:MethodInformation@<http://www.example.com/AA_Basis> ↙
    ↳ AND ?Tool1:Tool@<http://www.example.com/AA_Basis> AND ↙
    ↳ ?Analysis:ProcessInformation[uses_Method -> ?MethodInformation1] AND ↙
    ↳ ?Analysis[uses_LoadCase -> ?LoadCase] AND ?LoadCase[ist_Type -> ↙
    ↳ ?LoadCaseType] AND ?Analysis:Analysis@<http://www.example.com/02_Analysis> ↙
    ↳ AND ?Analysis[uses_AnalysisModel -> ↙
    ↳ ?AnalysisModel]@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?AnalysisModel:AnalysisModel@<http://www.example.com/02_Analysis> AND ↙
    ↳ ?AnalysisModel[AnalysisModel_based_on_ModelCreation -> ?ModelCreation] AND ↙
    ↳ ?ModelCreation:ModelCreation@<http://www.example.com/03_ModelCreation> AND ↙
    ↳ ?ModelCreation[uses_ProductInformation->?ProductInformation] AND ↙
    ↳ ?MethodInformation2[uses_Tool -> ?Tool2] AND ↙
    ↳ ?MethodInformation2:MethodInformation@<http://www.example.com/AA_Basis> ↙
    ↳ AND ?Tool2:Tool@<http://www.example.com/AA_Basis> AND ↙
    ↳ ?ModelCreation:ProcessInformation[uses_Method -> ?MethodInformation2] AND ↙
    ↳ ?LoadCase:LoadCase AND ?LoadCaseType:LoadCaseType AND ↙
    ↳ ?ProductInformation:ProductInformation.
```

Listing B.9: "Helper information chain"-Rule.

```
@{options[outorder(?ProductInformation, ?Datasource), fillNull, ↙
    ↳ EvaluationMethod(choose)]}
?-
?ProductInformation:ProductInformation[has_Datasource -> ?Datasource] AND NOT ↙
    ↳ _unify(?Datasource, DATENQUELLE_NotAvailable).
```

Listing B.10: "Not managed product information"-Query.

B.3. Ontology Schemata

Basis Schema This module consists of the classes that are imported by every other process module (*01_Assessment*, *02_Analysis* and *03_ModelCreation*) (*cf.* Figure 7.19). This way they do not have to be maintained at different locations.
In table B.1, the *AA_Basis* concepts (*cf.* Figure B.8) are explained in more detail:

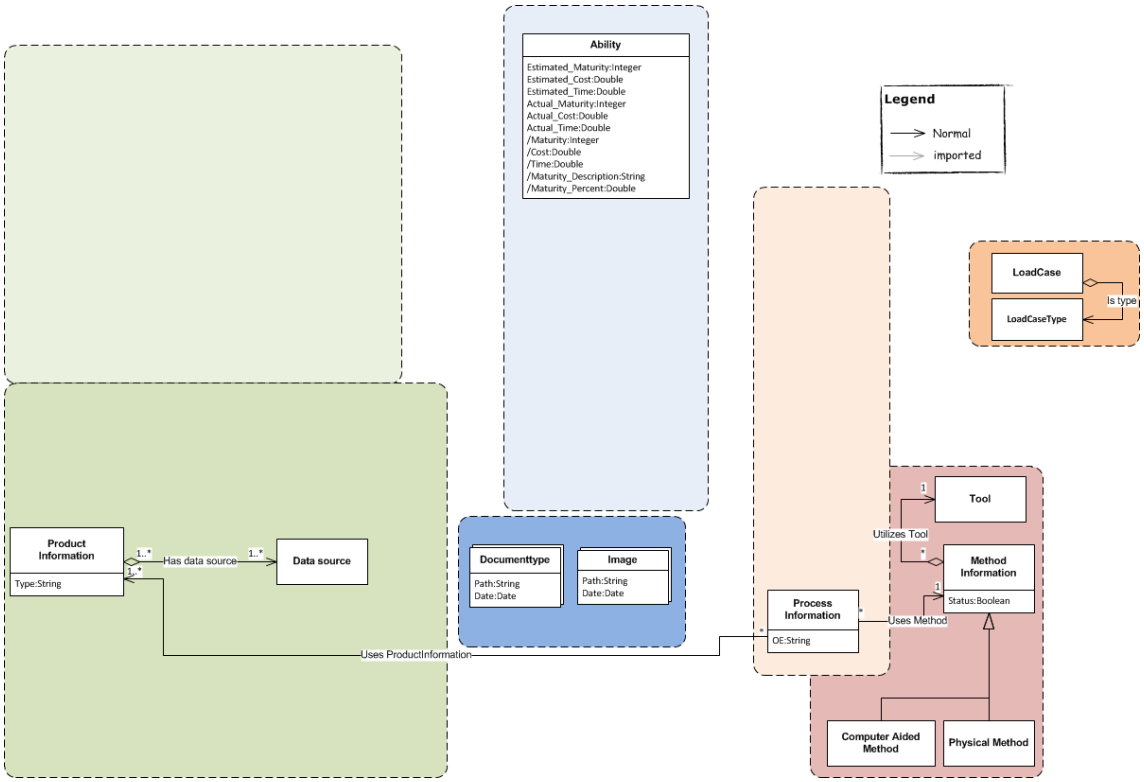


Figure B.8.: Basis Schema.

Concept	Description
`Image` & Subconcepts	Holds information, i.e., the path, name and date, about the images linked to various instances.
`DocumentType` & Subconcepts	Like the Image concept DocumentType stores information about dates, paths and names of source documents, like Excel files, associated with several instances.
`Property`	Properties describe the behavior of a product like functionalities, safety but also aesthetic aspects and costs as well as compatibilities like environmental compatibility.
`LoadCase`	The concept load case describes the specification of a scenario which is used during an analysis to investigate the product properties. An example for a test scenario is a “fishhook maneuver” of the US NCAP rating or an ISO “lane change manoeuvre”.
`LoadCaseType`	Classification of LoadCases

Continues on next page.

Concept	Description
Method-Information & Subconcepts	The concept method information describes the applied methods and tools during product development. To process information during the PDP, methods and tools are needed to transform input product information into output product information. All instances are linked to a tool.
Product-Information	The concept product information describes all pieces of information which are documented within product representations during the PDP.
Process-Information	The concept process information describes all process steps of an extended problem solution cycle.
Ability	The Ability is a kind-of association concept between processes and product information. The ability of a transformation of product information depends on the applied tools and methods during the process step and can be described in this context by three types of skills: • The costs of performing a process step with a distinct tool or method • The expenditure of time of performing a process step with a distinct tool or method • The quality of the output information of performing a process step with a distinct tool or method Each of these factors is either based on an actual or an estimated value. Often, a true value is not available and in this case, an estimated value is used. If the actual value can be determined in a later phase of the development, it can be added and is then used instead of the former estimate.
Tool	Tools used by methods.
DataSource	Software systems that manage the product information.

Table B.1.: *AA_Basis* concepts descriptions.

Assessment Schema This module consists of the information associated with the Process Assessment. It describes the connection between these Assessments and the related Properties (the process output) and the input in form an AnalysisResult (*cf.* Figure B.9).
Table B.2 explains the *Assessment* module's concepts in detail:

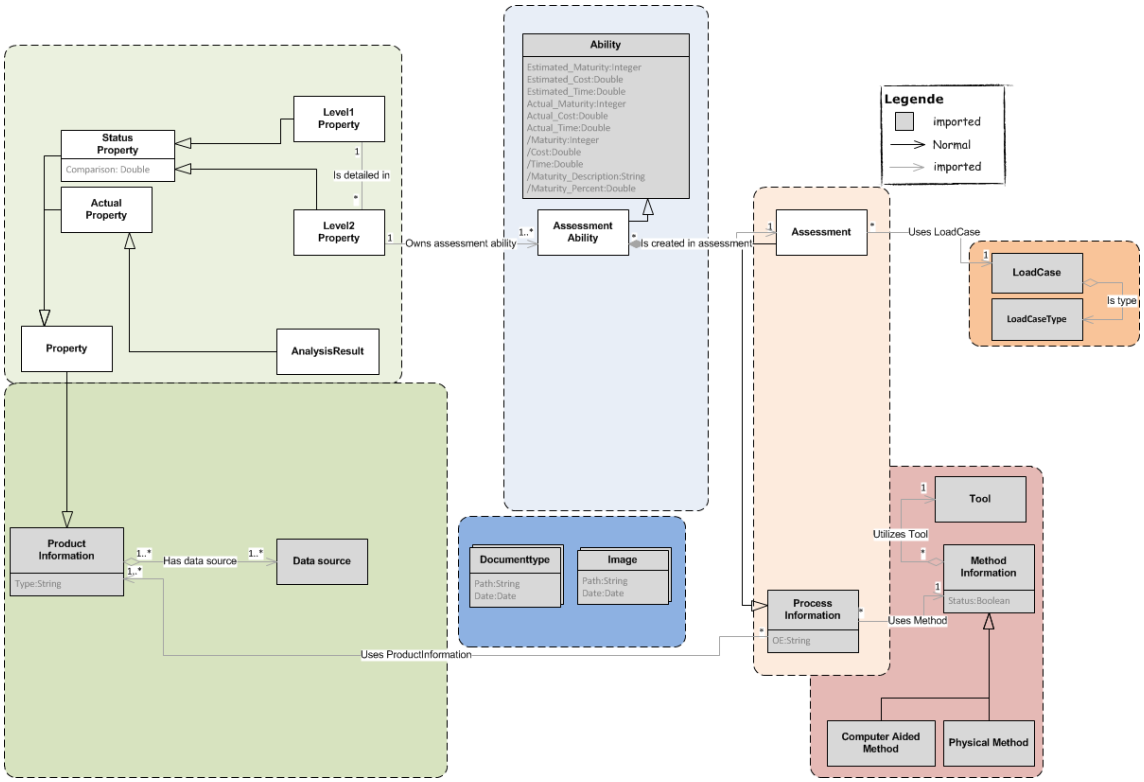


Figure B.9.: Assessment Schema.

Concept	Description
`Assessment`	The actual properties are compared with the target properties during assessment in order to calculate the status of the properties.
`ActualProperty`	Actual properties describe the currently existing properties for a developed car that can be determined by various calculations. This concept represents the parent concept of analysis results. At the current moment, ActualProperty does only have one child concept, but this will change in the future. Therefore, it has already been introduced.
`AnalysisResult`	Analysis results are the determined properties, which can be objectively measured during a trail or calculated during a simulation or subjectively estimated by an expert. An AnalysisResult is the input product information for an assessment.

Continues on next page.

Concept	Description
StatusProperty	The status of properties is calculated by the quotient between the actual and the target properties. The sum of all statuses is an indicator for the current achievement of the product development project.
Level-1-Property	The status properties are divided into different levels, representing different granularities. Level-1-properties own an attribute "Property relevance" which represents the importance if this property in comparison to the others. A higher number means less important.
Level-2-Property	The status properties are divided into different levels, representing different granularities. In this ontology, the first two levels have been modeled.
Assessment-Ability	The association concept between assessment and level-2-properties.

Table B.2.: *01_Assessment* concepts descriptions.

Analysis Schema This module consists of the information associated with the Process Analysis. It describes the connection between these Analyses and the related AnalysisResults (the process output) and the input in form of an AnalysisModel (*cf.* Figure B.10).
Table B.3 explains the concepts in detail:

Concept	Description
Analysis	The actual product properties are analysed (measured, estimated or calculated) during analysis (Weber 2005).
AnalysisAbility	The association concept between analysis and analysis result.
AnalysisModel	An analysis model is necessary to determine the product properties during the PDP. Models are abstractions of the reality and are used to investigate sub-aspects of the product. Examples for analysis models are hardware prototypes as well as simulation models like Multi Body Simulation (MBS) models or Software in the Loop (SiL) models.
AnalysisResult	See explanation in the previous section B.3.

Continues on next page.

Concept	Description

Table B.3.: *02_Analysis* concepts descriptions.

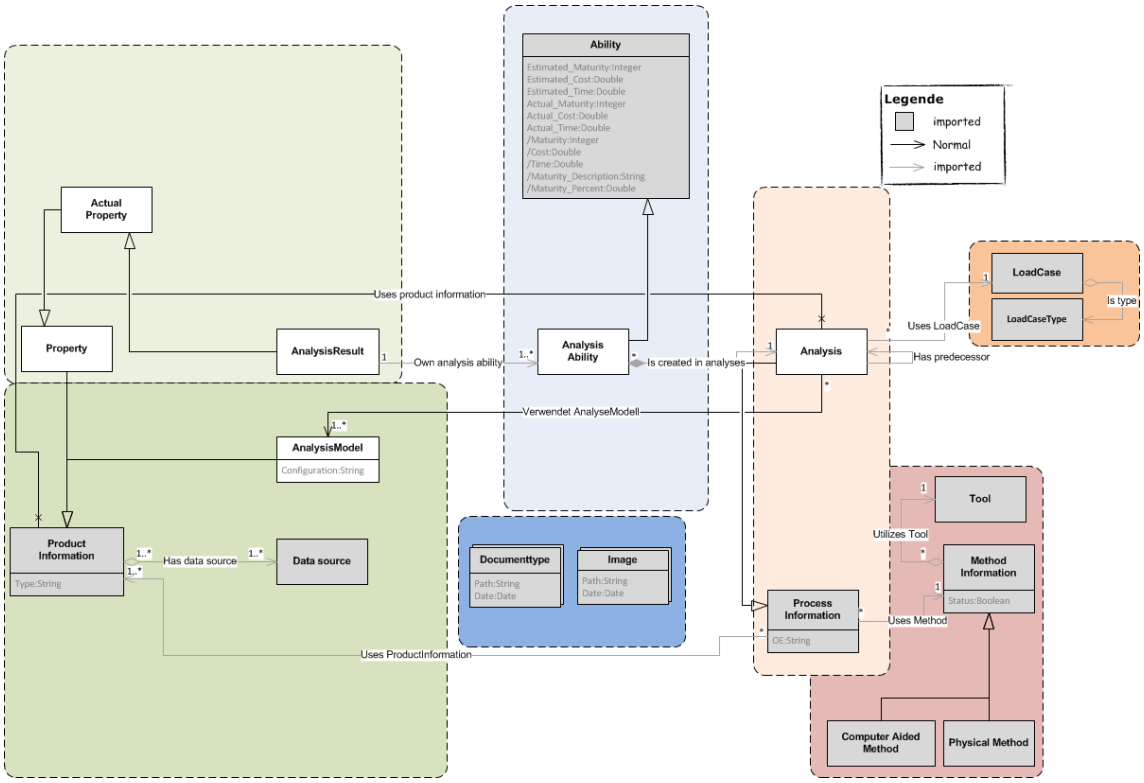


Figure B.10.: Analysis Schema.

ModelCreation Schema This module consists of the information associated with the `Process ModelCreation`. It describes the connection between these model creations and the related `AnalysisModels` (the process output) and the input in the form of `SolutionConcepts` (*cf.* Figure B.11). Table B.4 explains the concepts in detail:

Concept	Description
`AnalysisModel`	See explanation in the former section B.3.
`ModelCreation-Ability`	The association concept between AnalysisModel and ModelCreation.
`ModelCreation`	The product characteristics are filtered, preprocessed and assembled to an analysis model during the model creation step.

Continues on next page.

Concept	Description
`Characteristic` & `SolutionConcept`	Characteristics are the parameter of a product and describe the chosen solution concept. Only the characteristics can be directly influenced and adjusted by the product developer. Examples for characteristics are the shape, the structure or the material of a product (Weber 2005). SolutionConcepts describe the virtual or physical parts, geometries, functions or structures that are used to create an analysis model.
`Milestone`	SolutionConcepts are available beginning with a milestone in the PDP that is connected to a fixed date. Milestones also own the "maturity" attribute, which is a factor in calculations for the Planner Matrix.

Table B.4.: *03_ModelCreation* concepts descriptions.

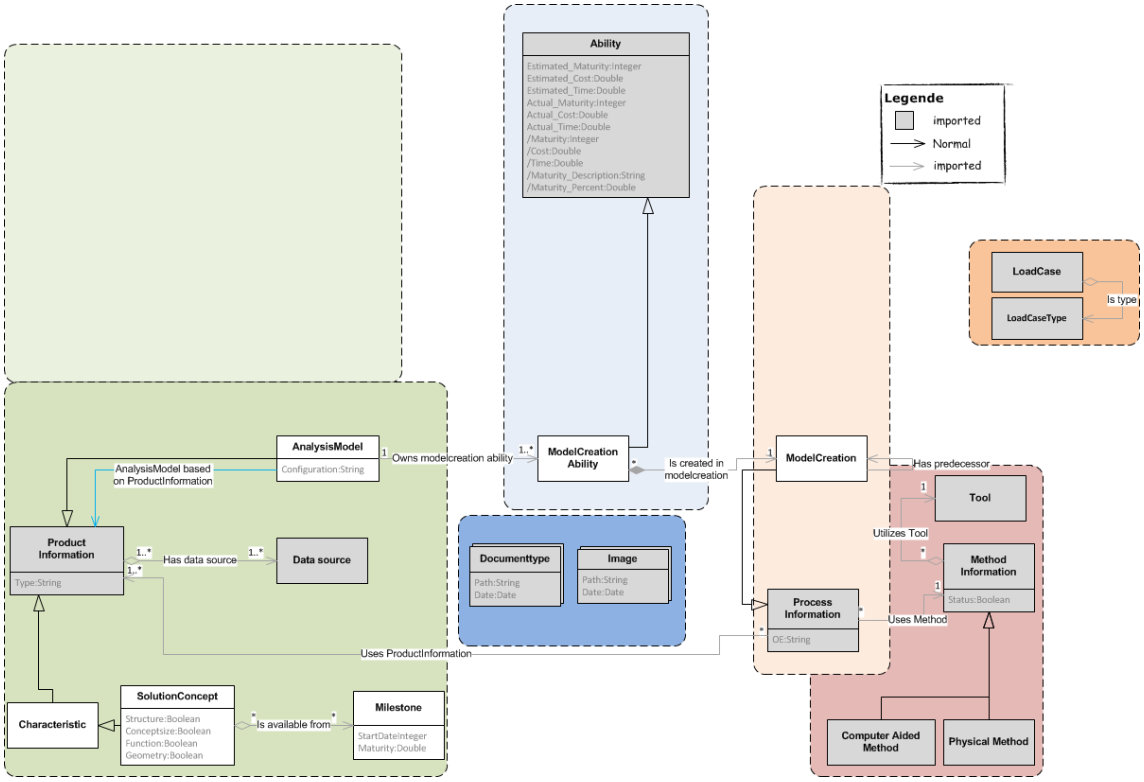


Figure B.11.: ModelCreation Schema.